# Computer Vision

## Computer Science Tripos Part II, Easter Term 1998

J G Daugman

1. Overview. Goals of computer vision; why they are so difficult.
2. Image sensing, pixel arrays, CCD cameras, framegrabbers.
3. Sampling theory. Finite differences and directional derivatives.
4. Filtering; convolution; correlation. 2D Fourier domain theorems.
5. Edge detection operators; the information revealed by edges.
6. The Laplacian and its zero-crossings. Logan's Theorem.
7. Scale-space, multi-resolution, causality. Wavelets.
8. Texture, colour, stereo, and motion descriptors. Disambiguation.
9. Lambertian and specular surfaces. Reflectance maps.
10. Bayesian inference in vision; knowledge-driven interpretations.
11. Infering shape from shading: surface geometry.
12. Boundary descriptors; Fundamental Theorem of Curves; codons.
13. Object-centered coords. Solid parameterization. Superquadrics.
14. Inverse problems; energy minimization, relaxation, regularization.
15. Model-based vision. Appearance-based versus volumetric models.
16. Applications and case studies. Face recognition.

# Computer Vision

## Computer Science Tripos Part II, Easter Term 1998

J G Daugman

## 1 Overview. Goals of computer vision; why they are so difficult.

Computer vision seeks to generate intelligent and useful descriptions of visual scenes and sequences, and of the objects that populate them, by performing operations on the signals received from video cameras.

Some examples of computer vision applications and goals:

- automatic face recognition, and interpretation of expression
- visual guidance of autonomous vehicles
- medical image analysis, interpretation, and diagnosis
- manufacturing: manipulation, grading, and assembly of parts
- OCR: recognition of printed or handwritten characters and words
- agriculture: visual grading and harvesting of produce
- smart offices: tracking of persons and objects; understanding gestures
- signature verification; other visual biometric-based identification
- visually endowed robotic helpers
- security monitoring and alerting
- intelligent interpretive prostheses for the blind
- tracking of moving objects; collision avoidance; stereoscopic depth
- object-based (model-based) compression of video streams
- general scene understanding

In many respects, computer vision is an "AI-complete" problem: building general-purpose vision machines would entail, or require, solutions to most of the general goals of artificial intelligence. It would require finding ways of building flexible and robust visual representations of the world, maintaining and updating them, and interfacing them with attention, goals and plans.

Like other problems in AI, the challenge of vision can be described in terms of building a *signal-to-symbol converter.* The external world presents itself only as physical signals on sensory surfaces (such as videocamera, retina, microphone...), which explicitly express very little of the information required for intelligent understanding of the environment. These signals must be converted ultimately into symbolic representations whose manipulation allows the machine or organism to interact intelligently with the world.

Although vision seems like such an effortless and immediate faculty for humans and other animals, it has proven exceedingly difficult to automate. Some of the reasons for this include the following:

1. An image is a two-dimensional optical projection, but the world we wish to make sense of visually is three-dimensional. In this respect, vision is *"inverse optics:"* we need to invert the 3D $\longrightarrow$ 2D projection in order to recover world properties (object properties in space); but the 2D $\longrightarrow$ 3D inversion of such a projection is, strictly, mathematically impossible.

   In another respect, vision is *"inverse graphics:"* graphics begins with a 3D world description (in terms of object and illuminant properties, viewpoint, etc.), and "merely" computes the resulting 2D image, with its occluded surfaces, shading and shadows, gradients, perspective, etc. Vision has to perform exactly the inverse of this process!

2. Very few visual tasks can be successfully performed in a purely data-driven way ("bottom-up" image analysis). Consider the next image examples: the foxes are well camouflaged by their textured backgrounds; the foxes occlude each other; they appear in several different poses and perspective angles; etc. How can there possibly exist mathematical operators for such an image that can:

- perform the figure-ground segmentation of the scene (into its objects and background)
- infer the 3D arrangements of objects from their mutual occlusions
- infer surface properties (texture, colour) from the 2D image statistics
- infer volumetric object properties from their 2D image projections
- and do all of this in "real time?" (This matters quite a lot in the natural world "red in tooth and claw," since survival depends on it.)

Consider now the actual image data of a face, shown as a pixel array with luminance plotted as a function of (X,Y) pixel coordinates. Can you see the face in this image, or even segment the face from its background, let alone recognize the face? In this form, the image reveals both the complexity of the problem and the poverty of the data.

This "counsel of despair" can be given a more formal statement:

Most of the problems we need to solve in vision are *ill-posed,* in Hadamard's sense that a *well-posed* problem must have the following set of properties:

- its solution exists;
- its solution is unique;
- its solution depends continuously on the data.

Clearly, few of the tasks we need to solve in vision are well-posed problems in Hadamard's sense. Consider for example the problems of:

- infering depth properties from an image
- infering surface properties from image properties
- infering colours in an illuminant-invariant manner
- infering structure from motion, shading, texture, shadows, ...
- interpreting the mutual occlusions of objects, and stereo disparity
- recognizing a 3D object regardless of its rotations about its three axes in space (e.g. a chair seen from many different angles)
- understanding an object that has never been seen before
- etc. ...

...but enough counsel of despair. Let us begin with understanding what an image array is.

## 2 Image sensing, pixel arrays, CCD cameras, framegrabbers.

A CCD video camera contains a dense array of independent sensors, which convert incident photons focused by the lens onto each point into a charge proportional to the light energy there. The local charge is "coupled" (hence CCD) capacitively to allow a voltage (V=Q/C) to be read out in a sequence scanning the array. The number of pixels (picture elements) is typically about 442,000 within a grid that is 1/3" x 1/3" in size, so each pixel sensor is a mere 13 microns across (0.013 mm).

Spatial resolution of the image is thus determined both by the density of elements in the CCD array, and by the properties of the lens which is forming the image. Luminance resolution (the number of distinguishable grey levels) is determined by the number of bits per pixel resolved by the digitizer, and by the inherent signal-to-noise ratio of the CCD array.

Colour information arises (conceptually if not literally) from three separate CCD arrays preceded by different colour filters, or mutually embedded as subpopulations within a single CCD array. It is encoded into the camera output either as a high-frequency "chrominance burst" (to be separately demodulated and decoded); or else signaled on a separate channel ("luma" and "chroma" portions of an S signal); or else provided as three separate RGB colour channels (red, green, blue).

A framegrabber contains a high-speed analogue-to-digital converter (ADC) which discretizes this video signal into a byte stream. Standard pixel array formats are:

- NTSC (North American standard): 640 x 480 pixels, at 30 frames/second (actually there is an interlace of alternate lines scanned out at 60 "fields" per second)
- PAL (European, UK standard): 768 x 576 pixels, at 25 frames/second (actually alternate lines are scanned out at 50 "fields" per second)

Note what a vast quantity of data a video sequence contains! 768 x 576 pixels/frame x 25 frames/second = 11 million pixels per second. Each pixel may be resolved to 8 bits in each of the three colour planes, hence 24 x 11 million = 264 million bits per second! How can we possibly cope with this data flux, let alone understand the objects and events creating such an image stream? ...

## 3 Sampling theory. Finite differences and directional derivatives.

Images are represented as rectangular arrays of numbers representing image intensities at particular locations. Each element of such an array is called a pixel, for picture element. A colour image may be represented in three separate such arrays called "colour planes," containing red, green, and blue components as monochromatic images.

The total number of independent pixels in an image array determines the spatial resolution of the image. Independent of this is the grey-scale (or colour) resolution of the image, which is determined by the number of bits of information specified for each pixel. These separate dimensions are illustrated in the following family of images, showing the effects of differing quantization accuracies for spatial and luminance information.

It is typical for a monochromatic ("black & white") image to have resolution of 8 bits/pixel. This creates 256 different possible intensity values for each pixel, from black (0) to white (255), with all shades of grey in between. A full-colour image may be quantized to this depth in each of the three colour planes, requiring a total of 24 bits per pixel. However, it is common to represent colour more coarsely or even to combine luminance and chrominance information in such a way that their *total* information is only 8 or 12 bits/pixel.

Because quantized image information is thus fundamentally discrete, the operations from calculus which we might want to perform on an image, like differentiation (to find edges) or integration (to perform convolutions or transforms), must be done in their discrete forms. The discrete form of a derivative is a *finite difference*. The discrete form of an integral is a (suitably normalized) *summation*. However, for the sake of conceptual familiarity, it is still commonplace in computer vision to represent such operations using their usual notations from continuous mathematics, with the understanding that the operation itself (as with everything else in Computer Science!) is of course discrete.

Finally, the task of parsing up an image into regions that correspond to object boundaries or property boundaries (called *segmentation,*) is represented by discrete set relations involving subsets of the image array. Thus a region of an image corresponding to an object or demarcating some image property is a subset of the array elements, and their boundary is a closed string of discrete coordinates. Operations such as "region-growing" and "prarie fires" seek to accrete pixels into sets forming ever larger domains as long as some local condition is satisfied (such as no edges encountered) or some local image property remains homogeneous (such as a textural signature or colour). Finding the largest possible subsets of pixels that parse the image into a small number of regions that each retain such properties, with refinements such as splitting and merging, is a generic approach to image segmentation.

The discreteness of image arrays imposes an upper limit on the amount of information they can contain. One way to describe this is by the total bit count, but this does not relate to the optical properties of image information. A better way is through Nyquist's Theorem, which tells us that the highest *spatial frequency* component of information contained within the image is equal to one-half the sampling density of the pixel array. (Intuitively, this is because at least two samples are required to signify a single cycle of a sinewave: its peak and its trough.) Thus, a pixel array containing 640 columns can represent spatial frequency components of image structure no higher than 320 cycles/image. For the same reason, if image frames are sampled in time at the rate of 30 per second, then the highest *temporal frequency* component of information contained in the image sequence is 15 Hertz.

## 4 Filtering; convolution; correlation. 2D Fourier domain theorems.

Nearly all image processing and feature encoding operations are based to some extent on the theory of Fourier Analysis.

Even if the operations never actually require computing a Fourier transform, their underlying principles and concepts (such as scale; edge or motion energy; filtering; directional derivative; textural signature; statistical structure; etc.) must be understood at least partially in "spectral" (i.e. Fourier) terms.

In addition to this explanatory role, Fourier analysis can be used directly to construct useful visual representations that are invariant under translation (change in position), rotation, and dilation (change in size). This is therefore the representation underlying many pattern classification and recognition applications, such as optical character recognition (OCR).

Finally, even many operations in pattern recognition that might not seem related in any way to Fourier analysis, such as computing correlations, convolutions, derivatives, differential equations, and diffusions, are much more easily implemented in the Fourier domain. (Powerful algorithms like the FFT make it easy to go back and forth rapidly between the image and Fourier domains).

For all of these reasons, we will review some principles and techniques of Fourier analysis with a view to understanding some of the basic operations in computer vision. Applications include edge detection operators, analysis of motion, texture descriptors, and wavelet-based feature detectors.

Consider an image as a black & white luminance distribution over the $(x, y)$ plane: a real-valued (indeed, a positive-valued) two-dimensional function $f(x, y)$.

Any image can be represented by a linear combination of basis functions:

$$f(x, y) = \sum_k a_k \Psi_k(x, y) \tag{1}$$

where many possible choices are available for the expansion basis functions $\Psi_k(x, y)$. In the case of Fourier expansions in two dimensions, the basis functions are the bivariate complex exponentials:

$$\Psi_k(x, y) = \exp(i(\mu_k x + \nu_k y)) \tag{2}$$

where the complex constant $i = \sqrt{-1}$. A complex exponential contains both a real part and an imaginary part, both of which are simple (real-valued) harmonic functions:

$$\exp(i\theta) = \cos(\theta) + i\sin(\theta) \tag{3}$$

which you can easily confirm by the power-series that define the transcendental functions such as exp, cos, and sin:

$$\exp(\theta) = 1 + \frac{\theta}{1!} + \frac{\theta^2}{2!} + \frac{\theta^3}{3!} + \cdots + \frac{\theta^n}{n!} + \cdots, \tag{4}$$

$$\cos(\theta) = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} + \cdots, \tag{5}$$

$$\sin(\theta) = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \cdots, \tag{6}$$

(It has been said that the most remarkable and far-reaching relationship in all of mathematics is the simple yet counterintuitive "Euler Relation" implied by Eqt (3) above: $e^{i\pi} + 1 = 0$, which also contains the five most important mathematical constants, and symbolizes the subject of harmonic analysis.)

Fourier Analysis computes the coefficients $a_k$ that yield an expansion of the image $f(x, y)$ in terms of complex exponentials:

$$f(x, y) = \sum_k a_k \exp(i(\mu_k x + \nu_k y)) \tag{7}$$

where the parameters $\mu_k$ and $\nu_k$ define the coordinates of the 2D Fourier domain. These $(\mu_k, \nu_k)$ coordinates are called vector spatial frequencies, and the array of them must span the $(\mu, \nu)$ Fourier plane in a uniform cartesian lattice.

It is often useful to think of the $(\mu, \nu)$ Fourier plane as resolved into polar coordinates, where $\omega = \sqrt{\mu^2 + \nu^2}$ is (scalar) <u>spatial frequency</u> and $\phi = \tan^{-1}(\nu/\mu)$ is (scalar) <u>orientation</u>.

Each Fourier coefficient $a_k$ is computed as the orthonormal projection of the entire image $f(x, y)$ onto the conjugate Fourier component $\exp(-i(\mu_k x + \nu_k y))$ associated with that coefficient:

$$a_k = (2\pi)^{-2} \int_X \int_Y \exp(-i(\mu_k x + \nu_k y)) f(x, y)\, dx\, dy \tag{8}$$

Note that these computed Fourier coefficients $a_k$ are <u>complex-valued.</u> To get a complete representation in the 2D Fourier domain for an image with $n$ x $n$ pixels, the number of $(\mu_k, \nu_k)$ vector frequency components whose associated coefficients $a_k$ must be computed is also $n$ x $n$.

### 4.1 Useful Theorems of 2D Fourier Analysis

Many important steps in computer vision such as feature extraction and invariant pattern recognition depend at least partly on a small set of Fourier theorems. We will review some main ones here, together with their direct consequences for practical computer vision applications. In every case, the input image is denoted $f(x, y)$, and its 2D Fourier transform (given by the set of computed coefficients $a_k$ spanning the Fourier plane) is denoted by $F(\mu, \nu)$.

**Shift Theorem:** Shifting the original pattern in $(x, y)$ by some 2D displacement $(\alpha, \beta)$ merely multiplies its 2DFT by $\exp(-2\pi i(\alpha\mu + \beta\nu))$. Thus the 2DFT of the shifted pattern $f(x-\alpha, y-\beta)$ is: $F(\mu, \nu)\exp(-2\pi i(\alpha\mu+\beta\nu))$.

**Practical Application:** The power spectrum of any isolated pattern is thus translation-invariant: it does not depend on where the pattern is located within the image, and so you don't have to find it first. The power spectrum is defined as the product of the pattern's 2DFT, $F(\mu, \nu)$, times its complex conjugate, $F^*(\mu, \nu)$, which just requires that the sign (–) of the imaginary part of $F(\mu, \nu)$ gets reversed. You can easily see that the power spectrum of the shifted pattern $f(x - \alpha, y - \beta)$, namely:

$$\exp(-2\pi i(\alpha\mu + \beta\nu))F(\mu, \nu)\exp(2\pi i(\alpha\mu + \beta\nu))F^*(\mu, \nu),$$

is equal to the power spectrum of the original unshifted pattern, namely: $F(\mu, \nu)F^*(\mu, \nu)$. Thus the power spectrum is translation-invariant.

**Similarity Theorem:** If the size of the original pattern $f(x, y)$ changes (shrinks/expands), say by a factor $\alpha$ in the $x$-direction, and by a factor $\beta$ in the $y$-direction, becoming $f(\alpha x, \beta y)$, then the 2DFT of the pattern, $F(\mu, \nu)$, also changes (expands/shrinks) by the reciprocal of those factors and with similarly scaled amplitude. It becomes: $\frac{1}{|\alpha\beta|}F(\frac{\mu}{\alpha}, \frac{\nu}{\beta})$.

**Rotation Theorem:** If the original pattern $f(x, y)$ rotates through some angle $\theta$, becoming $f(x\cos(\theta) + y\sin(\theta), -x\sin(\theta) + y\cos(\theta))$, then its 2DFT $F(\mu, \nu)$ also just rotates through the same angle. It becomes: $F(\mu\cos(\theta) + \nu\sin(\theta), -\mu\sin(\theta) + \nu\cos(\theta))$.

**Practical Application:** Size- and orientation-invariant pattern representations can be constructed by these relationships. Specifically, if the Fourier domain $(\mu, \nu)$ is now mapped into log-polar coordinates $(r, \theta)$

where $r = \log(\sqrt{\mu^2 + \nu^2})$ and $\theta = \tan^{-1}(\nu/\mu)$, then any dilation (size change) in the original pattern becomes simply a translation along the $r$-coordinate; and any rotation of the original pattern becomes simply a translation along the orthogonal $\theta$-coordinate in this log-polar Fourier domain. But we saw earlier that translations are made immaterial by taking a power spectrum, and so these effects of dilation and rotation of the pattern are eliminated in such a representation.

Combined with the translation-invariant property of the power spectrum, we now see how it becomes possible to represent patterns in a manner that is independent of their position in the image, their orientation, and their size (i.e. the Poincaré group of transformations) These principles are routinely exploited in machine optical character recognition; in military recognition of aircraft profiles; and in "optical computing" generally.

**Convolution Theorem:** Let function $f(x,y)$ have 2DFT $F(\mu,\nu)$, and let function $g(x,y)$ have 2DFT $G(\mu,\nu)$. The convolution of $f(x,y)$ with $g(x,y)$, which is denoted $f * g$, combines these two functions to generate a third function $h(x,y)$, whose value at location $(x,y)$ is equal to the integral of the product of functions $f$ and $g$ after one is flipped and undergoes a relative shift by amount $(x,y)$:

$$h(x,y) = \int_\alpha \int_\beta f(\alpha,\beta) g(x-\alpha, y-\beta)\, d\alpha d\beta \tag{9}$$

Thus, convolution is a way of combining two functions, in a sense using each one to blur the other, making all possible relative shifts between the two functions when computing the integral of their product to obtain the output as a 2D function of these amounts of shift.

Convolution is extremely important in vision because it is the basis for filtering. It is also the essential neural operation in the brain's visual cortex, where each neurone's receptive field profile is convolved with the retinal image. In the above integral definition, if the minus (−) signs were simply replaced with (+) signs, the new expression would be the correlation integral.

The Convolution Theorem states that convolving two functions $f(x,y)$ and $g(x,y)$ together in the image domain, simply multiplies their two 2DFT's together in the 2D Fourier domain:

$$H(\mu,\nu) = F(\mu,\nu) G(\mu,\nu) \tag{10}$$

where $H(\mu,\nu)$ is the 2DFT of the desired result $h(x,y)$.

This is extremely useful as it is much easier just to multiply two functions $F(\mu, \nu)$ and $G(\mu, \nu)$ together, to obtain $H(\mu, \nu)$, than to have to convolve $f(x, y)$ and $g(x, y)$ together (if the kernel is larger than tiny, say larger than about 5 x 5) to obtain $h(x, y)$. Of course, exploiting the Convolution Theorem means going into the 2D Fourier Domain and computing the 2DFT's of $f(x, y)$ and $g(x, y)$, and then performing yet another (inverse) FFT in order to recover $h(x, y)$ from the resulting $H(\mu, \nu)$. But with available powerful and fast 2D-FFT algorithms, this is very efficient.

**Practical Application: Filtering.** The starting-point of all feature extraction and image understanding operations is the filtering of an image $f(x, y)$ with some set of filters $g_k(x, y)$. Filtering is a linear operation implemented by the convolution of an image $f(x, y)$ with filter kernel(s) $g_k(x, y)$. The resulting output "image" $h_k(x, y)$ then normally undergoes non-linear operations of various kinds for image segmentation, motion detection, texture analysis, pattern recognition, and object classification.

The 2D discrete convolution of an image array with a 2D filter kernel can be represented algebraically in the following form, with the earlier continuous integrals being replaced by discrete summations:

$$\text{result}(i, j) = \sum_m \sum_n \text{kernel}(m, n) \cdot \text{image}(i - m, j - n)$$

The following short section illustrates a simple C program for implementing 2D convolution. Provided that the kernel is a small array, e.g. smaller than about 5 x 5, it is faster to perform a filtering operation explicitly in the image domain as illustrated below, rather than bothering with computing Fourier transforms in order to replace the convolution by multiplication of the transforms.

## Simple C program for performing image convolutions

In the following simple example, the array **image** is being convolved with the (typically much smaller) array **kernel**, in order to generate a new image array **result** as the output of the convolution. (Problems with array boundaries have been ignored here for simplicity.) Discrete convolution such as illustrated here is the key operation for all image processing and front-end stages of computer vision.

```
int  i, j, m, n, sum, image[iend][jend],
     kernel[mend][nend], result [iend][jend];

for (i = 0; i < iend; i++) {
    for (j = 0; j < jend; j++) {
        sum = 0;
        for ( m = 0; m < mend; m++) {
            for ( n = 0; n < nend; n++ ) {
                sum += kernel[m][n] * image[i-m][j-n];
            }
        }
        result[i][j] = sum/(mend*nend);
    }
}
```

If we chose to perform the convolution in the Fourier domain because the kernel array is large, then of the four nested **for loops** in the C code above, the inner two loops would be entirely eliminated. The innermost executable line inside the $i$ and $j$ **for loops** would simply be:

```
Result[i][j] = Kernel[i][j] * Image[i][j];
```

but the program would have to be preceded by FFTs of `kernel[i][j]` (trivial) and of `image[i][j]`, and followed by an FFT of `Result[i][j]`. Since the complexity of a 2D FFT is on the order of $n^2 \log_2(n)$ where $n^2$ is the number of pixels, plus $n^2$ multiplications in the innermost loop, the total complexity of the Fourier approach is $n^2(2\log_2(n)+1)$. In contrast, the number of multiplications in the explicit convolution above (not including all the array-addressing) is `iend*jend*mend*nend` (note that `iend*jend`$= n^2$). Hence you can calculate that the trade-off point occurs when the convolution kernel size `mend*nend` is about $\approx 2(\log_2(n)+1)$: a very small convolution kernel indeed, roughly 5 x 5 for a 512 x 512 image. For convolutions larger than this tiny one, the Fourier approach is faster.

**Differentiation Theorem:** Computing the derivatives of an image $f(x,y)$ is equivalent to multiplying its 2DFT, $F(\mu,\nu)$, by the corresponding frequency coordinate raised to a power equal to the order of differentiation:

$$\left(\frac{d}{dx}\right)^m \left(\frac{d}{dy}\right)^n f(x,y) \overset{2DFT}{\Longrightarrow} (2\pi i\mu)^m (2\pi i\nu)^n F(\mu,\nu) \qquad (11)$$

A particularly useful implication of this theorem is that isotropic differentiation, which treats all directions equally (for which the lowest possible order of differentiation is 2nd-order, known as the Laplacian operator $\nabla^2$) is equivalent simply to multiplying the 2DFT of the image by a paraboloid:

$$\nabla^2 f(x,y) \equiv \left(\frac{d^2}{dx^2} + \frac{d^2}{dy^2}\right) f(x,y) \stackrel{2DFT}{\Longrightarrow} -4\pi^2(\mu^2 + \nu^2)F(\mu,\nu) \qquad (12)$$

**Practical Application: Multi-Resolution Edge Detection. (next)**

## 5 Edge detection operators; the information revealed by edges.

Computer vision applications invariably begin with *edge detection*, be the edges straight, curvilinear, or closed boundary contours. There are several reasons why edges are important, and why detecting the edges in a scene can be regarded as an elementary form of constructing a *signal-to-symbol converter:*

- Edges demarcate the boundaries of objects, or of material properties.
- Objects have parts, and these are typically joined with edges.
- The three-dimensional distribution of objects in a scene usually generates occlusions of some objects by other objects, and these form occlusion edges which reveal the geometry of the scene.
- Edges can be generated in more abstract domains than luminance. For example, if some image property such as colour, or a textural signature, or stereoscopic depth, suddenly changes, it forms a highly informative "edge" in that domain.
- Velocity fields, containing information about the trajectories of objects, can be organized and understood by the movements of edges. (The motions of objects in space generates velocity discontinuities at their edges.)
- The central problem of stereoscopic 3D depth vision is the "correspondence problem:" matching up corresponding regions of two images from spatially displaced cameras. Aligning edges is a very effective way to solve the correspondence problem. The same principle applies to measuring velocities (for image frames displaced in time, rather than displaced in space) by tracking edges to align corresponding regions and infer velocity (ratio of object displacement to temporal interval).

In summary, DISCONTINUITIES = INFORMATION.

An intuitive way to find edges is to compute the derivative of a (1D) signal, since this will be large where the luminance is changing rapidly. Since image arrays are discrete, we must use the *finite difference* representation of a derivative, and this is implemented by a *convolution*: If our (1D) luminance array is L[n] as indicated below by the schematic pixel values, then the first-order finite difference operator (h[0],h[1])=(1,-1) when convolved with L[n] would generate an output which is only large in amplitude where L[n] has edges:

However, note an important disadvantage of this approach: "rightward edges" (say, from dark to bright) generate the opposite sign from "leftward edges" (say, from bright to dark). We would prefer to generate the same detection signal regardless of the *polarity* of the edge.

A solution is to convolve the discrete luminance data L[n] instead with the *second finite difference operator*, defined as (h[-1],h[0],h[1])=(1,-2, 1) and look for the *zero-crossings* of this operator. These correspond to peaks or troughs of the first finite difference operator that we considered above, and thus they reveal the edges, regardless of their polarity.

In the two-dimensional case, we have the choice of using *directional derivative* operators, or *non-directional* ones. An example of a directional operator is one which integrates (sums) pixels in one direction, but differentiates (differences) them in the perpendicular direction. Clearly, such an operator will detect edges only in a specific orientation; - namely the orientation along which the integration was done. A example of such a directional edge detector is the following 3 x 3 array:

In comparison, an *isotropic* operator such as the Laplacian (sum of second derivatives in two perpendicular orientations) has no preferred direction; that is the meaning of isotropy. It will detect edges in all orientations. The picture at the top of the next page illustrates such an effect. A good discrete approximation to the Laplacian operator is the following 3 x 3 array:

Notice how each of these simple 3 x 3 operators sums to zero when all of their elements are combined together. These types of operators (of which there are obviously numerous other examples, differing in array sizes as well as element composition) are called *filters*, because of their spectral consequences for favouring some spatial frequency bands and orientations at the expense of others. Their zero-sum property means that they are insensitive to the overall brightness value of a scene, as we would desire: they have "no DC term." (Their Fourier transform is equal to zero at the origin.) They also may, or may not, have a certain preferred, or characteristic *direction*; a certain phase or *symmetry* (even or odd); and a certain *scale*, defined by the spacing between changes of sign in the elements in (larger) arrays.

Illustration of edge-detection by convolution with an isotropic Laplacian operator, and marking the zero-crossings of the result of the convolution.

Edges in images are defined at different scales: some transitions in brightness are gradual, others very crisp. More importantly, at different scales of analysis, different edge structure emerges.

Example: an image of a leopard that has been low-pass filtered (or analyzed at a coarse scale) has edge outlines corresponding to the overall form of its body. At a somewhat finer scale of analysis, image structure may be dominated by the contours of its "spots." At a still finer scale, the relevant edge structure arises from the texture of its fur.

In summary, non-redundant structure exists in images at different scales of analysis (or if you prefer, in different frequency bands).

The basic recipe for extracting edge information from images is to use a multiscale family of image filters (convolution kernels). A wide variety of these are in standard use, differing in terms such as:

- isotropic (circularly symmetric), or anisotropic (directional)
- self-similar (dilates of each other), or not self-similar
- separable (expressible as product of two 1D functions), or not
- degree of conjoint uncertainty in the information resolved
- size of support (number of "taps," or pixels, in the kernel)
- preferred non-linear outputs (zero-crossings; phasor moduli; energy)
- theoretical foundations (e.g. Logan's Theorem)

## 6 The Laplacian $\nabla^2 G_\sigma(x, y) * I(x, y)$ and its zero-crossings. Logan's Theorem.

One highly influential idea due to Marr (1981), that is pervasively exploited today in front-end machine vision systems, is to convolve the image with a multi-scale family of isotropic (non-directional) blurred 2nd-derivative filters, and to retain only their output zero-crossings. These correspond well to the edges in the image, at each scale.

The primary motivation for doing this comes from Logan's Theorem (1977) concerning the "richness" of Laplacian zero-crossings for band-limited signals. What Logan proved (albeit only in the 1D case) is that subject to two constraints, the zero-crossings alone suffice to represent the signal completely (i.e. it could be perfectly recovered from just its zeros, up to a scale factor).

This is a truly remarkable result. Consider the fact that a signal is continuous and dense, but in any finite interval it will have only a finite (countable) number of zero-crossings (e.g., 7). How can those 7 points completely determine what the signal does everywhere else within this finite interval??

The two constraints are:

1. The signal must be strictly bandlimited to one octave, or less. This means that its highest frequency component must be no more than twice its lowest frequency component.
   (This constraint is much more powerful than it may appear.)

2. The signal must have no complex zeros in common with its Hilbert Transform. This effectively excludes purely amplitude-modulated signals. For example, a pure sinewave whose amplitude is merely modulated will have exactly the same zero-crossings as the unmodulated sinusoid, so their zero-crossings would not distinguish between them. Thus AM signals cannot be represented by zero-crossings.

The $\nabla^2 G_\sigma(x, y)$ filter kernel that is convolved with the image serves to bandpass-filter it. In the 2D Fourier domain, as we have seen, the spectral consequence of the Laplacian operator $\nabla^2 \equiv \left(\frac{d^2}{dx^2} + \frac{d^2}{dy^2}\right)$ is to multiply the image spectrum by a paraboloid: $(\mu^2 + \nu^2)$. Clearly this emphasizes the high frequencies at the expense of the low frequencies, and eliminates the DC component entirely (hence the output is centered around a mean of zero).

Blurring the Laplacian by a Gaussian $G_\sigma(x, y)$ of scale $\sigma$, simply limits the high-frequency components. The 2DFT of a Gaussian is also a Gaussian, with reciprocal dimension (by the Similarity Theorem discussed earlier). The scale parameter $\sigma$ determines where the high-frequency cut-off occurs.

The resulting bandwidth of a $\nabla^2 G_\sigma(x, y)$ filter is about 1.3 octaves, regardless of what value for scale parameter $\sigma$ is used. Note that this doesn't *quite* satisfy the first constraint of Logan's Theorem.

Note also that by commutativity of linear operators, the order in which these steps are applied to the image $I(x, y)$ doesn't matter. First computing the Laplacian of the image, and then blurring the result with the Gaussian, is equivalent to first convolving the image with the Gaussian and then computing the Laplacian of the result:

$$\nabla^2 \left[G_\sigma(x, y) * I(x, y)\right] = G_\sigma(x, y) * \nabla^2 I(x, y) \tag{13}$$

Moreover, both of these sequences are equivalent to just convolving the image with a <u>single</u> filter kernel, namely the Laplacian of a Gaussian: $\left[\nabla^2 G_\sigma(x, y)\right] * I(x, y)$. Clearly this is the preferred implementation, since it just involves a single convolution.

Some open theoretical issues in this approach are:

1. It is not clear how to generalize the constraint of one-octave bandlimiting to the case of 2D signals (images). E.g. should their 2DFT be confined to an annulus in the Fourier plane, whose outer radius is twice its inner radius?; or to four squares in the four quadrants of the Fourier plane that satisfy the one-octave constraint on each frequency axis? The first method doesn't work, and clearly the second filter is no longer isotropic!

2. Whereas the zeros of a 1D signal (soundwave) are denumerable [countable], those of a 2D signal (image) are not. Rather, they form "snakes" that are continuous contours in the plane.

3. As a practical matter, the $\nabla^2 G_\sigma(x, y) * I(x, y)$ approach to edge extraction tends to be very noise-sensitive. Many spurious edge contours appear that shouldn't be there. This defect inspired the development of more sophisticated non-linear edge detectors, such as Canny's, which estimates the local image signal-to-noise ratio (SNR) to adaptively optimize its local bandwidth. This, however, is very computationally expensive.

4. Finally, strong claims were originally made that $\nabla^2 G_\sigma(x,y) * I(x,y)$ edge-detecting filters describe how human vision works. In particular, the receptive field profiles of retinal ganglion cells were said to have this form. However, counterexamples reveal several visual tasks that humans are able to perform, effortlessly and pre-attentively, which we could not perform if our visual systems functioned in this way.

## 7 Scale-space, multi-resolution, causality. Wavelets.

Images contain information at multiple scales of analysis, so detecting visual features (such as edges) must be done across a range of different scales.

- An interesting property of edges as defined by the zero-crossings of multi-scale operators whose scale is determined by convolution with a Gaussian, is that as the Gaussian is made coarser (larger), new edges (new zero-crossings) can never appear. They can only merge and thus become fewer in number. This property is called causality. It is also sometimes called 'monotonicity,' or 'the evolution property,' or 'nice scaling behaviour.'

- One reason why causality is important is that it ensures that features detected at a coarse scale of analysis were not spuriously created by the blurring process (convolution with a low-pass filter) which is the normal way to create a multi-scale image pyramid using a hierarchy of increasing kernel sizes. One would like to know that image features detected at a certain scale are "grounded" in image detail at the finest resolution.

- For purposes of edge detection at multiple scales, a plot showing the evolution of zero-crossings in the image after convolution with a linear operator, as a function of the scale of the operator which sets the scale (i.e. the width of the Gaussian), is called scale-space.

- Scale-space has a dimensionality that is one greater than the dimensionality of the signal. Thus a 1D waveform projects into a 2D scale-space. An image projects into a 3D scale space, with its zero-crossings (edges) forming surfaces that evolve as the scale of the Gaussian changes. The scale of the Gaussian, usually denoted by $\sigma$, creates the added dimension.

- A mapping of the edges in an image (its zero-crossings after such filtering operations, evolving with operator scale) is called a scale-space fingerprint. Several theorems exist called "fingerprint theorems" showing that the Gaussian blurring operator uniquely possesses the property of causality. In this respect, it is a preferred edge detector when combined with a bandpass or differentiating kernel such as the Laplacian.

- However, other non-linear operators have advantageous properties, such as reduced noise-sensitivity and greater application for extracting features that are more complicated (and more useful) than mere edges.

### 7.1 2D Gabor "Logons;" Quadrature Pair Wavelets

The family of filters which uniquely achieve the lowest possible conjoint uncertainty (i.e. minimal dispersion, or variance) in both the space domain and the Fourier domain are the complex exponentials multiplied by Gaussians. These are sometimes known as Gabor wavelets, or "logons." In one dimension:

$$f(x) = \exp(-i\mu_0(x - x_0)) \exp(-(x - x_0)^2/\alpha^2)$$

This is a Gaussian localized at position $x_0$, complex modulated at frequency $\mu_0$, and with size or spread constant $\alpha$. It is noteworthy that such wavelets have Fourier Transforms $F(\mu)$ with exactly the same functional form, but with their parameters merely interchanged or inverted:

$$F(\mu) = \exp(-ix_0(\mu - \mu_0)) \exp(-(\mu - \mu_0)^2\alpha^2)$$

Note that for the case of a wavelet $f(x)$ centered on the origin so $x_0 = 0$, its Fourier Transform $F(\mu)$ is simply a Gaussian centered on the modulation frequency $\mu = \mu_0$, and whose width is $1/\alpha$, the reciprocal of the wavelet's space constant. This shows that it acts as a bandpass filter, passing only those frequencies that are within about $\pm\frac{1}{\alpha}$ of the wavelet's modulation frequency $\mu_0$.

Dennis Gabor (1946) named these wavelets "logons" from the Greek word for information, or order: logōs. Because of the optimality of such wavelets under the Uncertainty Principle, Gabor proposed using them as an expansion basis to represent signals. In particular, he wanted them to be used in broadcast telecommunications for encoding continuous-time information. He called them the "elementary functions" for a signal. Unfortunately, because such functions are mutually non-orthogonal, it is very difficult to obtain the actual coefficients to be used with the elementary functions in order to expand a given signal in this basis. (Gabor himself could not solve this problem, although he went on to invent holography and to win the Nobel Prize in Physics in 1974.)

When a family of such Gabor functions are parameterized to be self-similar, i.e. they are dilates and translates of each other so that they all have a common template ("mother" and "daughter"), then they constitute a (non-orthogonal) *wavelet basis.* Today it is known that infinite classes of wavelets exist which can be used as the expansion basis for signals. Because of the self-similarity property, this amounts to representing or analyzing a signal at different scales.

This general field of investigation is called *multi-resolution analysis*, and we have already encountered its importance for extracting edge features. For these reason, today wavelets are all the rage in computer vision research.

### 7.2 Generalization of Wavelet Logons to 2D for Image Analysis

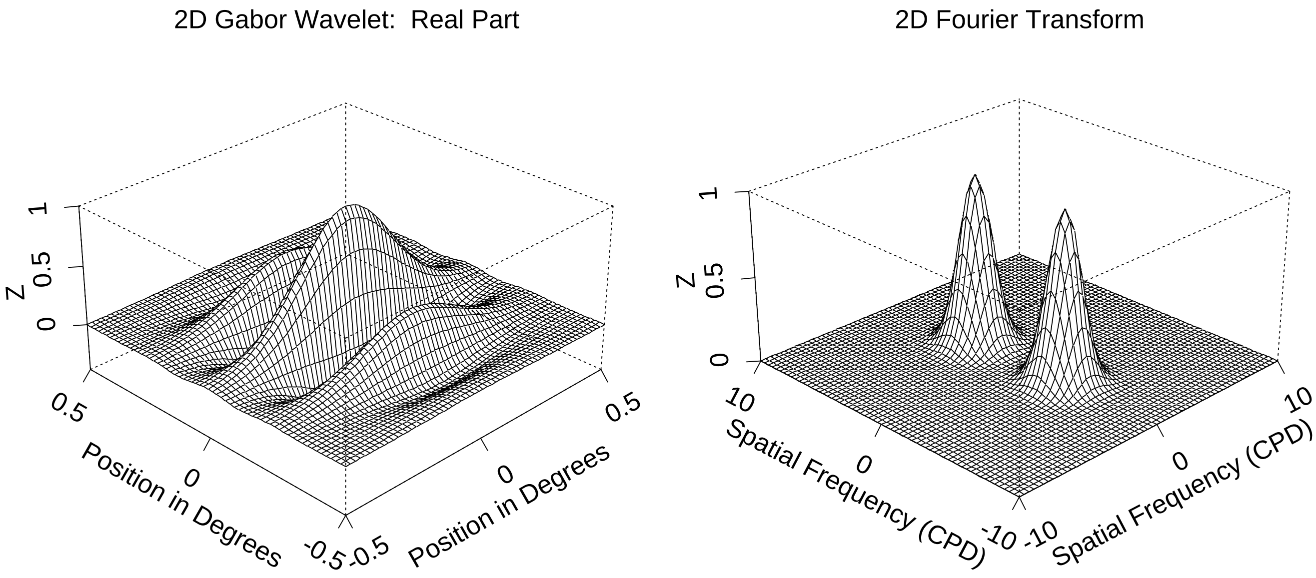


Figure 1: The real part of a 2D Gabor wavelet, and its 2D Fourier transform.

An effective method for extracting, representing, and analyzing image structure is the computation of the 2D Gabor wavelet coefficients for the image. This family of 2D filters were originally proposed as a framework for understanding the orientation-selective and spatial-frequency-selective receptive field properties of neurones in the brain's visual cortex, as well as being useful operators for practical image analysis problems. These 2D filters are conjointly optimal in extracting the maximum possible information both about the orientation and modulation of image structure ("what"), simultaneously with information about 2D position ("where"). The 2D Gabor filter family uniquely achieves the theoretical lower bound on joint uncertainty over these four variables in the Uncertainty Principle when it is suitably generalized.

These properties are particularly useful for texture analysis because of the 2D spectral specificity of texture as well as its variation with 2D spatial position. These wavelets are also used for motion detection, stereoscopic vision, and many sorts of visual pattern recognition such as face recognition. A large and growing literature now exists on the efficient use of this non-orthogonal expansion basis and its applications.

Two-dimensional Gabor wavelets have the functional form:

$$f(x,y) = e^{-\left[(x-x_0)^2/\alpha^2+(y-y_0)^2/\beta^2\right]} e^{-i[u_0(x-x_0)+v_0(y-y_0)]}$$

where $(x_0, y_0)$ specify position in the image, $(\alpha, \beta)$ specify effective width and length, and $(u_0, v_0)$ specify modulation, which has spatial frequency $\omega_0 = \sqrt{u_0^2 + v_0^2}$ and direction $\theta_0 = \arctan(v_0/u_0)$. (A further degree-of-freedom not included above is the relative orientation of the elliptic Gaussian envelope, which creates cross-terms in $xy$.) The 2D Fourier transform $F(u,v)$ of a 2D Gabor wavelet has exactly the same functional form, with parameters just interchanged or inverted:

$$F(u,v) = e^{-\left[(u-u_0)^2\alpha^2+(v-v_0)^2\beta^2\right]} e^{-i[x_0(u-u_0)+y_0(v-v_0)]}$$

The real part of one member of the 2D Gabor filter family, centered at the origin $(x_0, y_0) = (0, 0)$ and with unity aspect ratio $\beta/\alpha = 1$ is shown in the Figure, together with its 2D Fourier transform $F(u,v)$.

By appropriately parameterizing them for dilation, rotation, and translation, 2D Gabor wavelets can form a complete self-similar (but non-orthogonal) expansion basis for images. If we take $\Psi(x,y)$ to be some chosen generic 2D Gabor wavelet, then we can generate from this one member a complete self-similar family of 2D wavelets through the generating function

$$\Psi_{mpq\theta}(x,y) = 2^{-2m}\Psi(x', y')$$

where the substituted variables $(x', y')$ incorporate dilations in size by $2^{-m}$, translations in position $(p, q)$, and rotations through orientation $\theta$:

$$x' = 2^{-m}[x\cos(\theta) + y\sin(\theta)] - p$$

$$y' = 2^{-m}[-x\sin(\theta) + y\cos(\theta)] - q$$

It is noteworthy that as consequences of the similarity theorem, shift theorem, and modulation theorem of 2D Fourier analysis, together with the rotation isomorphism of the 2D Fourier transform, all of these effects of the generating function applied to a 2D Gabor mother wavelet $\Psi(x,y) = f(x,y)$ have corresponding identical or reciprocal effects on its 2D Fourier transform $F(u,v)$. These properties of self-similarity can be exploited when constructing efficient, compact, multi-scale codes for image structure.

The completeness of 2D Gabor wavelets as an expansion basis for any image is illustrated in the above reconstruction of a facial image, in stages. Note how efficiently the facial features, such as the eyes and mouth, are represented using only a handful of the wavelets. Later we will see how this can be exploited both for automatic feature localization, and for face recognition.

**Reconstruction of Lena: 25, 100, 500, and 10,000 Two-Dimensional Gabor Wavelets**

Figure 2: Illustration of the completeness of 2D Gabor wavelets as basis functions.

### 7.3 Unification of Domains

Until now we have viewed "the image domain" and "the Fourier domain" as very different domains of visual representation. But now we can see that the "Gabor domain" of representation actually embraces and unifies both of these other two domains. How?

In the wavelet equations above, the scale constant $\alpha$ (and $\beta$ in the 2D case) actually builds a continuous bridge between the two domains. If the scale constant is set very large, then the Gaussian term becomes just 1 and so the expansion basis reduces to the familiar Fourier basis. If instead the scale constant is made very small, then the Gaussian term shrinks to a discrete delta function (1 only at the location $x = x_0$, and 0 elsewhere), so the expansion basis implements pure space-domain sampling: a pixel-by-pixel image domain representation. This allows us to build a continuous deformation between the two domains when representing, analyzing, and recognizing image structure, merely by changing a single scaling parameter in this remarkable, unifying, expansion basis.

## A "philosophical" comment about 2D Gabor wavelets.

Aristotle defined vision as "knowing what is where." We have noted the optimality (conjoint uncertainty minimization) property of 2D Gabor wavelets in the two domains for extracting structural ("what") and positional ("where") information. Thus if we share Aristotle's goal for vision, then we cannot do better than to base computer vision representations upon these wavelets. Perhaps this is why mammalian visual systems appear to have evolved their use; the receptive field profiles of isolated neurones in the brain's visual cortex, as determined by the spatial distribution of excitatory and inhibitory inputs to each so-called "simple cell," can be well-described as quadrature-paired 2D Gabor wavelets. At the present time, this is more-or-less the standard model for how the brain's visual cortex represents the information in the retinal image.

### 7.4 Detection of Facial Features by Quadrature Gabor Wavelet Energy

One illustration of a practical application of such image operators is in the automatic localization of facial features. Interestingly, most facial features themselves can be described by just a handful of wavelets, since such features are after all just localized undulations having certain positions, orientations, spatial frequencies, and phases. By taking the modulus (sum of the squares of the real and imaginary parts) of a facial image after convolving it with complex-valued 2D Gabor wavelets, key facial features (eyes and mouth) are readily detected. This capability is illustrated in the Figure below.

Figure 3: Illustration of Facial Feature Detection by Quadrature Filter Energy. Left panel: original image. Right panel (clockwise from top left): the real part after 2D Gabor wavelet convolution; the imaginary part; the modulus energy; and this energy superimposed on the original (faint) image, illustrating successful feature localization.

## 8 Texture, colour, stereo, and motion descriptors. Disambiguation.

Many seemingly disparate tasks in computer vision actually share a common formal structure: to convert ill-posed, insoluble problems of inference from raw data, into well-posed problems in which we can compute object properties disambiguated from the image-formation processes which confound the raw luminance data itself.

One obvious aspect of this issue is the fact that images are 2D projections of 3D data which could, in principle, arise equally well from many different constellations of worlds and objects. A more subtle aspect is the fact that the information received as an image is the compound product of several factors that are difficult to disambiguate: (1) the nature, geometry, and wavelength composition of the illuminant(s); (2) properties of the objects imaged, such as: spectral reflectances; surface shape; surface albedo; surface texture; geometry, motion, and rotation angle; and (3) properties of the camera (or viewer), such as (i) geometry and viewing angle; (ii) spectral sensitivity; (iii) prior knowledge, assumptions, and expectations. The aim of this lecture is to study how these many factors can be disambiguated and even exploited, in order to try to make objective inferences about object and world properties from these ambiguous and confounded image properties.

### 8.1 Texture information.

Most surfaces are covered with texture of one sort or another. Texture can serve not only as a helpful identifying feature, but more importantly as a cue to surface shape because of the foreshortening it undergoes as it follows the shape of the object <u>if</u> one can assume that it has some uniform statistics along the surface itself. The following patterns illustrate the inference of surface slant and of 3D surface shape from texture cues when they are combined with the assumption of texture uniformity on the surface itself:

Texture is also a useful cue to image segmentation by parsing the image into local regions which are relatively homogeneous in their textural properties. Here are some illustrations:

How can one measure something as ill-defined as a "textural signature?" What is texture, anyway?

As implied by the root of the word, which links it to textiles, texture is defined by the existence of certain statistical *correlations* across the image. These can be almost anything, from quasi-periodic undulations as one might see in water ripples or in woven fabrics, to repetitive but highly punctate features. Many natural scenes, such as woodlands, grasslands, mountain ranges and other terrains, have such properties which give them a distinctive identifying visual signature. The unifying notion in all of these examples is *quasi-periodicity*, or repetitiveness, of some features.

The detection of quasi-periodicity is best done by Fourier methods. There are deep and multi-faceted links between many topics in statistics (such as time-series analysis, correlation, moments) and Fourier analysis. These links arise from the fact that the eigenfunctions of the Fourier transform, complex exponentials (sinusoids in quadrature), are of course periodic but also have a specific scale (frequency) and direction (wavefront). Thus they excel in detecting the existence of a correlation distance and direction, and in estimating the relative "power" represented in various components of quasi-periodic correlated structures.

Unfortunately, these eigenfunctions are globally defined, but we wish to use local regional information as a basis for texture-based image segmentation. Hence the ideal solution is to "window" the sinusoids so that they analyze the image characteristics only within a local region and thus extract the spectral statistics as a function that varies with location. The optimal set of windowing functions are bivariate Gaussians, since their joint spatial/spectral localization is greater than that of any other function. The product of complex exponentials times bivariate Gaussians are called *2D Gabor wavelets*, and these form a complete basis for image analysis and representation. The pictures below illustrate successful segmentation of collages of textured natural scenes, as well as of textured artificial objects, using such 2D wavelets for local spectral analysis to infer and measure their textural discriminators.

### 8.2 Colour information.

Colour is a nearly ubiquitous property of surfaces. Just like texture, it can serve both in object identification and in scene segmentation. But the fundamental difficulty in using the wavelength composition of images to infer the colour properties ("spectral reflectances") of objects, is the fact that the wavelengths received depend as much upon the *illuminant* as upon the spectral reflectances of the surface that is scattering back the light. When a yellow banana is illuminated in bluish light, the image that it forms obviously has a very different wavelength composition than when it is illuminated in reddish light. The central mystery of human colour perception is the fact that the banana still appears yellow ("colour constancy"). In computer vision, how can we possibly achieve this same vital capability of inferring an inherent underlying *object property* from a confounded (i.e., illuminant-wavelength dependent) set of *image properties*?

To give the problem a slightly more formal presentation:

- Let $I(\lambda)$ represent the wavelength composition of the illuminant (i.e. the amount of energy it contains as a function of wavelength $\lambda$, across the visible spectrum from about 400 nanometers to 700 nm).
- Let $O(\lambda)$ represent the inherent spectral reflectance of the object at a particular point: the fraction of incident light that is scattered back from its surface there, as a function of the incident light's wavelength $\lambda$.
- Let $R(\lambda)$ represent the actual wavelength mixture received by the camera at the corresponding point in the image of the scene.

Clearly, $R(\lambda) = I(\lambda)O(\lambda)$. The problem is that we wish to infer the "object colour" (its spectral reflectance as a function of wavelength, $O(\lambda)$), but we only know $R(\lambda)$, the actual wavelength mixture received by our sensor. So unless we can measure $I(\lambda)$ directly, how could this problem of inferring $O(\lambda)$ from $R(\lambda)$ possibly be solved?

One simple idea that has been proposed is to try actually to measure $I(\lambda)$ directly, by searching for highly specular (shiny, metallic, glassy) regions in an image where the reflected light might be a fairly faithful copy of $I(\lambda)$. This might be a glint from someone's glasses or from a shiny doorknob. Then at all other points in the image we need only to divide the $R(\lambda)$ we receive there by our other specular "measurement" of $I(\lambda)$, and we can then compute the desired $O(\lambda)$ across the image.

Clearly, this method has several weakness: (1) there may be no specular surfaces in the image; (2) those that there are may themselves affect somewhat the wavelength composition that they reflect (e.g. metals which have a brassy colour); and (3) the method is neither robust nor stable, since global inferences about scene interpretation depend critically upon uncertain measurements at (what may be just) a single tiny point in the image.

A more stable and interesting approach was developed by Dr E Land, founder of Polaroid, and is called the Retinex because he regarded it as modelled after biological visual systems (RETINa + cortEX). Land's critical observation was that (contrary to almost universal popular belief), the colour perceived in an area of a scene is *not* determined by the wavelength composition of light received from that area (!). A simple experiment proves this: illuminate a scene, such as a bowl of fruit containing (say) a yellow banana, a red tomato and a green pepper, with three different narrowband light sources, each of which contains a different wavelength (say red, green, or blue) and with adjustable intensities. (No other light sources are present.)

The first observation is that even under drastic changes in the intensities of each of the three illuminators, the objects maintain exactly their normal colours. Obviously the wavelength mixture reaching the eye from each object is drastically changing, in proportion to the illuminators, but there are no changes in perceived colours. The phenomenon does not depend upon knowing the natural colours for objects identifiable by (say) their shape; a collage of patches of coloured paper cut into random shapes, forming a *mondrian,* produces exactly the same effect.

The second observation is that even when the wavelength composition of light reflected from each object is exactly the same (i.e. the three light sources are adjusted separately for each object to ensure that the light reflected in the three wavebands as measured by a spectral photometer is exactly the same for each of the objects individually), they *still* retain their natural colours. The banana still looks yellow, the tomato still looks red, and the pepper still looks green, even when each one is sending *identical* wavelength "messages" to your eyes. This is rather miraculous.

The Retinex algorithm attempts to account for this remarkable biological phenomenon, and to provide a means to achieve similar *colour constancy* in computer vision systems so that they may "discount the illuminant" and infer the spectral reflectance properties of objects, independent of the composition of their illumination. Only a cursory description of Retinex will be given here.

The key idea is that *the colours of objects or areas in a scene are determined by their surrounding spatial context.* A complex sequence of ratios computed across all the boundaries of objects (or areas) enables the illuminant to be algebraically discounted in the sense shown in the previous Figure, so that object spectral reflectances $O(\lambda)$ which is what we perceive as their colour, can be infered from the available retinal measurements $R(\lambda)$ without explicitly knowing $I(\lambda)$.

### 8.3 Stereo information

Important information about depth can be obtained from the use of two (or more) cameras, in the same way that humans achieve stereoscopic depth vision by virtue of having two eyes. Objects in front or behind of the point in space at which the two optical axes intersect (as determined by the angle between them, which is controlled by camera movements or eye movements), will project into different relative parts of the two images. This is called *stereoscopic disparity.*

This "error signal" becomes greater in proportion to the distance of the object in front or behind the point of fixation, and so it can be calibrated to obtain a depth cue. It also becomes greater with increased spacing between the two eyes or cameras, since that is the "base of triangulation." (That is why the German Army in WWI introduced V-shaped binocular "trench periscopes" to increase stereoscopic visual acuity, for breaking camouflage by increasing the *effective* spacing between the viewer's two eyes to almost a meter.)

The essence of making use of such stereoscopic disparity cues is the need to solve the Correspondence Problem. In order to infer that the cylinder is in a different position relative to the background objects in the two frames shown, it is first necessary to detect the correspondence of the background objects in the two frames, or at least of their edges. This puts the two frames "into registration," so that the disparity of the foreground object can be detected.

Unfortunately, current algorithms for solving the Correspondence Problem tend to require very large searches for matching features under a large number of possible permutations. It is difficult to know which set of features in the two frames to select for comparison in evaluating the degree of alignment, when trying to find that relative registration which generates maximum correlation between the two background scenes.

One helpful approach here is to use a "multi-scale image pyramid," which steers the search in a coarse-to-fine fashion to maximize its efficiency. If one begins with very blurred copies of the original pair of image frames, then all of their information can be captured by a very small set of sparsely distributed pixels (since pixels closer together are so highly correlated by the large degree of blur). In such sparsely sampled coarse images, the permutation-matching space of possible corresponding points is greatly attenuated (factorially!) compared with full-resolution images.

After an adequate alignment match is found for low-resolution (blurred) copies of the image pair, the process repeats on somewhat higher resolution (less blurred) copies of the image pair but over a search space that has been greatly curtailed by having first found the coarse-scale solution. Such "pyramid" processes usually increment in one-octave steps (factors of two in improved resolution), from coarse to fine, spanning a total of perhaps four or five levels before the final solution is determined to within single-pixel precision.

Once the Correspondence Problem has thereby been solved, the inference of depth from object disparity in the two image frames is then just a matter of triangulation and "look-up" from a calibration table which includes information about the spacing between the two cameras (or eyes) and their focal lengths. (See the simplifying diagram on the previous page, for the case that the two cameras' optical axes are parallel and hence converged at infinity.) Specifically, if the two cameras have focal length $f$ and the optical centres of their lenses (remember the trench periscopes!) are separated by a distance $b$, and the disparity in the projections of some object point onto the two images (in opposite directions relative to their optical axis) is $\alpha$ in one image and $\beta$ in the other image, then the distance $d$ to the object in front of the two lenses (what we wanted to know!) is simply:

$$d = fb/(\alpha + \beta)$$

### 8.4 Motion information

Only a few vision applications actually involve just static image frames. That is basically vision "off-line;"– but the essence of an effective visual capability must be for real-time use in a dynamic environment. This requires the ability to detect and measure motion, and to be able thereby to draw inferences quickly (such as time-to-collision).

In a formal sense, the problem of computing motion information from an image sequence is very similar to that of computing stereo information.

- For stereo vision, we need to solve the Correspondence Problem for two images simultaneous in time but acquired with a spatial displacement.
- For motion vision, we need to solve the Correspondence Problem for two images coincident in space but acquired with a temporal displacement.
- The object's spatial "disparity" that can be measured in the two image frames once their backgrounds have been aligned, can be calibrated to reveal motion information when compared with the time interval, or depth information when compared with the binocular spatial interval.

Among the challenging requirements of motion detection and inference are:

1. Need to infer 3D object trajectories from 2D image motion information.
2. Need to make *local* measurements of velocity, which may differ in different image regions in complex scenes with many moving objects. Thus, a velocity vector field needs to be assigned over an image.

3. Need to disambiguate object motion from contour motion, so that we can measure the velocity of an object regardless of its *form*.
4. Need to measure velocities regardless of the size of the viewing aperture in space and in time (the spatial integration and temporal integration windows).
5. It may be necessary to assign more than one velocity vector to any given local image region (as occurs in "motion transparency")
6. We may need to detect a *coherent* overall motion pattern across many small objects or regions separated from each other in space.

The major classes of models and approaches to motion detection are:

**INTENSITY GRADIENT MODELS .**

Assume that the local time-derivative in image intensities at a point, across many image frames, is related to the local spatial gradient in image intensities because of object velocity $\vec{v}$:

$$-\frac{\partial I(x,y,t)}{\partial t} = \vec{v} \cdot \vec{\nabla} I(x,y,t) \tag{14}$$

Then the ratio of the local image time-derivative to the spatial gradient is an estimate of the local image velocity (in the direction of the gradient).

**DYNAMIC ZERO-CROSSING MODELS .**

Measure image velocity by first finding the edges and contours of objects (using the zero-crossings of a blurred Laplacian operator!), and then take the time-derivative of the Laplacian-Gaussian-convolved image:

$$-\frac{\partial}{\partial t}\left[\nabla^2 G_\sigma(x,y) * I(x,y,t)\right] \tag{15}$$

in the vicinity of a Laplacian zero-crossing. The amplitude of the result is an estimate of speed, and the sign of this quantity determines the direction of motion relative to the normal to the contour.

**SPATIO-TEMPORAL CORRELATION MODELS .**

Image motion is detected by observing a *correlation* of the local image signal $I(x,y,t)$ across an interval of space and and after an interval of time $\tau$. Finding the pair of these intervals which maximizes the correlation between $I(x,y,t)$ and $I(x-v_x\tau, y-v_y\tau, t-\tau)$ determines the two components of image velocity $v_x$ and $v_y$ that we desire to know.

**SPATIO-TEMPORAL SPECTRAL MODELS .**

It is possible to detect and measure image motion purely by *Fourier* means. This approach exploits the fact that motion creates a covariance in the spatial and temporal *spectra* of the time-varying image $I(x, y, t)$, whose three-dimensional (spatio-temporal) Fourier transform is defined:

$$F(\omega_x, \omega_y, \omega_t) = \int_X \int_Y \int_T I(x, y, t) e^{-i(\omega_x x + \omega_y y + \omega_t t)} dx dy dt \tag{16}$$

In other words, local image motion has a 3-D spectral consequence: the local 3-D spatio-temporal spectrum, rather than filling up 3-space $(\omega_x, \omega_y, \omega_t)$, collapses onto a 2-D inclined plane which includes the origin. Motion detection then occurs just by filtering the image sequence in space and in time, and observing that tuned spatio-temporal filters whose center frequencies are **co-planar** in this 3-space are activated together. This is a consequence of the **SPECTRAL CO-PLANARITY THEOREM**:

## 9 Lambertian and specular surfaces. Reflectance maps.

How can we infer information about the surface reflectance properties of objects from raw measurements of image brightness? This is a more recondite matter than it might first appear, because of the many complex factors which determine how (and where) objects scatter light.

Some definitions of surface type and properties:

- Surface albedo refers to the fraction of the illuminant that is re-emitted from the surface in all directions, in total. Thus, albedo corresponds more-or-less to "greyness."

  The amount of light reflected is the product of two factors: the albedo of the surface, times a geometric factor that depends on angle.

- A Lambertian surface is "pure matte." It reflects light equally well in all directions.

  Examples of Lambertian surfaces include snow, non-glossy paper, ping-pong balls, magnesium oxide...

  A Lambertian surface looks equally bright in all directions; the amount of light reflected depends only on the angle of incidence.

  If you looked inside a "Lambertian bottle" with an arbitrarily complex shape, illuminated with a point source of light from any angle, you could never infer the interior shape! It would have uniform brightness everywhere regardless of its actual shape.

- A specular surface is locally mirror-like. It obeys Snell's law (i.e. the angle of incidence of light is equal to the angle of reflection from the surface), and does not scatter light. Most metallic surfaces are specular.
- The reflectance map is a function $\phi(i,e,g)$ which relates intensities in the image to surface orientations of objects. It specifies the fraction of incident light reflected per unit surface area, per unit solid angle, in the direction of the camera; thus it has units of flux/steradian. It is a function of three variables (see previous Figure): $i$ is the angle of the illuminant, relative to the surface normal $N$; $e$ is the angle of a ray of light re-emitted from the surface; and $g$ is the angle between the emitted ray and the illuminant.

There are many types of reflectance functions, each of which is characteristic of certain surfaces and imaging environments. For a Lambertian surface, the reflectance function $\phi(i,e,g) = \cos(i)$ . It looks equally bright viewed from all directions; the amount of reflected light depends only on angle of illumination.

For surfaces such as the dusty surface of the moon, the reflectance function $\phi(i,e,g)$ depends only upon the ratio of the cosines of the angles of incidence and emission: $\cos(i)/\cos(e)$, but not upon their relative angle $g$ nor upon the surface normal $N$. In case you ever wondered, this is why the moon looks like a penny rather than a sphere. Even though the moon is illuminated by a point source (the sun), it does not fade in brightness towards its limbs (as $N$ varies). Surfaces with this property are called *lunar* surfaces.

For a specular surface, the reflectance function $\phi(i,e,g)$ is especially simple: $\phi(i,e,g) = 1$ when $i = e$ and both are coplanar with the surface normal $N$, so $g = i + e$ (Snell's law for a pure mirror); and $\phi(i,e,g) = 0$ otherwise.

Typically, surfaces have both specular and matte properties. For example, facial skin may vary from lunar (powdered) to specular (oily). The main purpose of powdering one's face is to specify $s$ and $n$ in this expression:

$$\phi(i,e,g) = \frac{s(n+1)(2\cos(i)\cos(e) - \cos(g))^n}{2} + (1-s)\cos(i) \qquad (17)$$

The first term is the specular component, and the second term is the matte component. $s$ is the fraction of light emitted specularly, and $n$ represents the sharpness of the specular peak. For glossy paint, typically the exponent $n$ may be about 20. (Obviously as $n$ grows very large, the exponentiated trig function approaches a delta function representing Snell's law: a very sharp power function of angle.)

Typically there is not just one point source of illumination, but rather a multitude of sources (such as the extended light source provided by a bright overcast sky). In a cluttered scene, much of the light received by objects has been reflected from other objects (and coloured by them...) One needs almost to think of light not in terms of ray-tracing but in terms of thermodynamics: a "gas" of photons in equilibrium inside a room.

Clearly, the only way to infer the nature and geometry of surface properties from image properties, given all of these complications in the way that surfaces reflect and scatter light, is to build in certain assumptions about the nature of the surfaces from other kinds of evidence. This requires us to consider the general problem of inference and integration of evidence (next topic).

## 10 Bayesian inference in vision; knowledge-driven interpretations.

It is virtually impossible to perform most computer vision tasks in a purely "bottom-up" fashion. Consider the following images, and how impoverished are the data which must support the task of object recognition!

An important "AI" perspective on vision is that vision is knowledge-driven. In this view, all of the front-end image processing is merely a distraction, if not an irrelevancy. What is really needed for vision is not a lot of theorems involving the 2D Fourier transform of the Laplacian of a Gaussian filter, but rather a good interface to an expert system that stores and indexes knowledge about such things as Dalmatian hounds and the general way that dogs behave when following a scent...

This section reviews the basic ideas behind Bayesian inference, which is a method fundamental to probability theory, statistics, and machine learning. Its purpose is to provide a means for integrating *prior* information (such as general knowledge about the sorts of things that populate the world, their properties and relationships, the metaphysics of objects, etc...) with empirical information gathered from incoming image data. This principle is expressed in the form of a basic rule for relating conditional probabilities in which the "antecedent" and "consequent" are interchanged. The value of this method for computer vision is that it provides a framework for continually updating one's theory of what one is looking at, by integrating continuously incoming evidence with the best available inference or interpretation so far.

### 10.1 Decisions under uncertainty.

Most real-world tasks (whose solution requires intelligence) involve degrees of uncertainty. Decision-making under uncertainty is especially characteristic in computer vision. The sources of uncertainty may include:

- the nature of the data or signals available
- the inherent problem of classifying or recognizing them
- the unpredictability of the future
- the uncertainty of causation
- the fact that objects and events have probabilities
- the fact that associative knowledge is probabilistic
- the inherent incompleteness or imperfection of processing
- possible undecidability of a problem, given all available data
- the "ill-posed" nature of many tasks
- inherent trade-offs such as speed versus accuracy

But despite these realities, decisions are required.

Statistical decision theory is the study of how to optimize certain measures of performance, given the available data and the decision environment as specified by costs/benefits, a-priori knowledge, speed and confidence requirements.

Examples of decisions-under-uncertainty in computer vision:

- Medical diagnosis; radiology: Is this a tumor? Does the cost of a possible False Alarm (taking a biopsy, frightening the patient unnecessarily) exceed the cost of possibly missing an early diagnosis? What should you do if the odds are 99% that it is just a benign cyst; but if it is a tumour, missing it now could be fatal?
- Military decision-making: a plane is seen approaching your aircraft carrier very low on the horizon and at high speed. Is it friend or foe? How should the costs of the two possible types of error (shooting down one of your own planes, vs allowing the whole aircraft carrier to be sunk) be balanced against their relative probabilities when making your decision?

Finally, how can decision strategies be updated by the integration of evidence that comes in gradually over time? How should a-priori knowledge about the probabilities of events in the world be combined with available incoming data?

## The Bayesian View

A highly influential formalism for integrating prior knowledge about the world (beliefs being expressed in terms of probabilities) with new incoming data (e.g. an image sequence), is that of the Bayesians.

Bayesianism interprets probability as "degree-of-belief," rather than as "frequency of occurence," and argues for weighing all evidence with all possible (imaginable) interpretations and their associated (estimated) probabilities. Baye's rule, named after the 17th-century cleric, Thomas Bayes, is a formalism for combining *prior knowledge or beliefs* with empirical observations. It is at once a theory of explanation, a procedure for the integration of evidence, and a protocol for decision-making.

We begin with an informal statement of Bayes' rule, and an illustration. If $x$ represents "state of the world" (e.g. the truth about the object being imaged) and $y$ represents "data" (e.g. the available image data), then the explanatory conditional probabilities $p(x|y)$ and $p(y|x)$ are related to each other and to their unconditional likelihoods $p(x)$ and $p(y)$ as follows:

$$p(x|y) = \frac{p(x)p(y|x)}{p(y)} \tag{18}$$

For example, a human agricultural expert (or an AI expert system) may have knowledge of the form $p(y|x)$: Given a plant (or a disease state) $x$, there is a corresponding conditional probability $p(y|x)$ of observing image data $y$. However, the goal of computer vision and pattern recognition is typically precisely to calculate the *inverse* conditional probability: given image data $y$, what is the probability $p(x|y)$ that this is plant $x$ (or disease state $x$, etc.)?

Bayes' rule (Eqt. 18) specifies the formal procedure for calculating such inferences $p(x|y)$, given the observations, the unconditional probabilities, and the prior expert agricultural knowledge $p(y|x)$. It thereby offers a clean and simple interface between a knowledge base and visual data. Indeed, as we shall see, it offers an interface that can be recursively applied for the continual upgrading of knowledge or beliefs as new visual data arrives incrementally.

### 10.2 Beliefs, probabilities, and Bayesian inference (material not lectured)

Most of probability theory was laid down by theologians: Blaise PASCAL (1623-1662) who gave it the axiomatization that we accept today; and Thomas BAYES (1702-1761) who expressed one of its most important and widely-applied propositions relating conditional probabilities.

Probability Theory rests upon two rules:

$\boxed{\text{Product Rule:}}$

$$\begin{aligned} p(A,B) &= \text{“joint probability of } \textit{both } A \textit{ and } B\text{”} \\ &= p(A|B)p(B) \\ & \text{or equivalently,} \\ &= p(B|A)p(A) \end{aligned}$$

Clearly, in case $A$ and $B$ are *independent* events, they are not conditionalized on each other and so

$$\begin{aligned} P(A|B) &= P(A) \\ \text{and } P(B|A) &= P(B), \end{aligned}$$

in which case their joint probability is simply $P(A,B) = P(A)P(B)$.

$\boxed{\text{Sum Rule:}}$

If event $A$ is conditionalized on a number of other events $B$, then the total probability of $A$ is the sum of its joint probabilities with all $B$:

$$p(A) = \sum_B p(A,B) = \sum_B p(A|B)p(B)$$

From the Product Rule and the symmetry that $p(A,B) = p(B,A)$, it is clear that $p(A|B)p(B) = p(B|A)p(A)$. Bayes' Theorem then follows:

$\boxed{\text{Bayes' Rule:}}$

$$p(B|A) = \frac{p(A|B)p(B)}{p(A)}$$

The importance of Bayes Rule is that it allows us to reverse the conditionalizing of events, and to compute $p(B|A)$ from knowledge of $p(A|B)$, $p(A)$, and $p(B)$. Often these are expressed as *prior* and *posterior* probabilities, or as the conditionalizing of hypotheses upon data.

Worked Example:

Suppose that a dread disease affects 1/1000th of all people. If you actually have the disease, a test for it is positive 95% of the time, and negative 5% of the time. If you don't have the disease, the test is positive 5% of the time. We wish to know how to interpret test results.

Suppose you test positive for the disease. What is the likelihood that you actually have it?

We use the above rules, with the following substitutions of "data" $D$ and "hypothesis" $H$ instead of $A$ and $B$:

$D$ = data: the test is positive
$H$ = hypothesis: you have the disease
$\bar{H}$ = the other hypothesis: you do not have the disease

Before acquiring the data, we know only that the *a priori* probability of having the disease is .001, which sets $p(H)$. This is called a *prior*. We also need to know $p(D)$.

From the Sum Rule, we can calculate that the *a priori* probability $p(D)$ of testing positive, whatever the truth may actually be, is:

$$p(D) = p(D|H)p(H) + p(D|\bar{H})p(\bar{H}) = (.95)(.001) + (.05)(.999) = .051$$

and from Bayes' Rule, we can conclude that the probability that you actually have the disease given that you tested positive for it, is much smaller than you may have thought:

$$p(H|D) = \frac{p(D|H)p(H)}{p(D)} = \frac{(.95)(.001)}{(.051)} = \boxed{0.019} \qquad \text{(less than 2\%)}.$$

This quantity is called the *posterior probability* because it is computed after the observation of data; it tells us how likely the hypothesis is, given what we have observed. (Note: it is an extremely common human fallacy to confound $p(H|D)$ with $p(D|H)$. In the example given, most people would react to the positive test result by concluding that the likelihood that they have the disease is .95, since that is the "hit rate" of the test. They confound $p(D|H) = .95$ with $p(H|D) = .019$, which is what actually matters.)

A nice feature of Bayes Theorem is that it provides a simple mechanism for repeatedly updating our assessment of the hypothesis as more data continues to arrive. We can apply the rule recursively, using the latest *posterior* as the new *prior* for interpreting the next set of data. In AI, this feature is important because it allows the systematic and real-time construction of interpretations that can be updated continuously as more data arrive in a time series, such as a flow of images or spoken sounds that we wish to understand.

### 10.3 Statistical Decision Theory

The Bayesian view focuses on the use of *priors*, which allow vision to be steered heavily by one's *a priori* knowledge about the world and the things which populate it. For example, probabilistic priors can express the notion that some events, objects, or interpretations are vastly more probable than others; that matter cannot just disappear, but does routinely become occluded; that objects rarely change their surface colour; that uniform texturing on a complex surface shape is a more likely interpretation than highly non-uniform texturing on a simple or planar shape; that a rigid rotation in three dimensions is a "better explanation" for deforming boundaries (if consistent with same) than actual boundary deformations in the object itself; and so forth. Being able to integrate formally such learned or even "metaphysical" assumptions about the world is one way in which Bayesian inference facilitate a "top-down" or AI-oriented, expert-system-oriented, approach to vision.

However, in many vision tasks, there may be no useful (or strong) priors. We may need to solve pattern recognition problems purely on the basis of some vector of acquired features from a given object or image; the task is to decide whether or not this feature vector is consistent with membership in a particular class or object category. In this sense, the problem of object identification amounts to a "same / different" decision between the presenting feature vector and one (or more) characteristic class feature vectors, even if we don't have any useful priors about the relative likelihoods of the possible object classes or interpretations.

The degree of match between two feature vectors must be computed and formally evaluated to make a decision of "same" or "different." Almost always, there is some similarity between "different" patterns, and some dissimilarity between "same" patterns. This creates a <u>decision environment</u> with four possible outcomes:

1. Hit: Actually same; decision "same".
2. Miss: Actually same; decision "different".
3. False Alarm: Actually different; decision "same".
4. Correct Reject: Actually different; decision "different".

We would like to maximize the probability of outcomes 1 and 4, because these are correct decisions. We would like to minimize the probability of outcomes 2 and 3, because these are incorrect decisions ("Type II" and "Type I" errors).

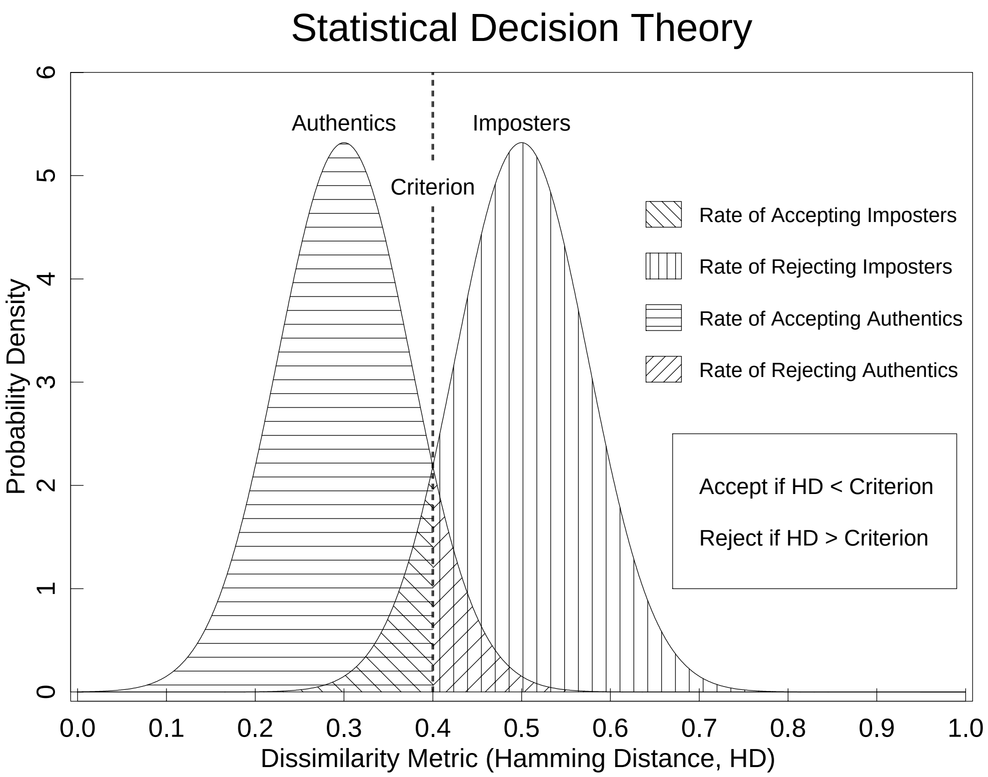


Figure 4: Two-choice Decision Environment of Statistical Decision Theory

We can adjust our decision threshold (become more liberal or more conservative) to reflect the costs and benefits of the four possible outcomes. But adjusting the decision threshold has coupled effects on the four outcomes:

- Increasing the "Hit" rate will also increase the "False Alarm" rate.
- Decreasing the "Miss" rate will also decrease the "Correct Reject" rate.

How can we understand these relationships in a theoretical formalism? How can we optimize the decision-making process?

During WWII, a theoretical framework was developed for understanding such decision environments in the context of radar. It was developed at the University of Michigan and became known as Signal Detection Theory, or also as Statistical Decision Theory. The signals received about the "state of the world" are modelled as arising from two noisy probability distributions. In the pattern recognition context, these two correspond to the class-object relationships of "same" versus "different." (In the schematic diagram above, the terms "authentic" and "imposter" were used.)

When a decision of "same" or "different" is made, based upon the observed similarity and some acceptability threshold, the probabilities of the four possible outcomes can be computed as the four areas lying under these two distributions to either side of the decision criterion. These four probabilities correspond to the shaded areas in the diagram. The computed error probabilities can be directly translated into a "confidence level" that we can assign to any decision that is made in this formalism. The result of being "liberal" or "conservative" in our decision-making is revealed in the ROC curve, for Receiver Operating Characteristic (the name is derived from radar analysis).

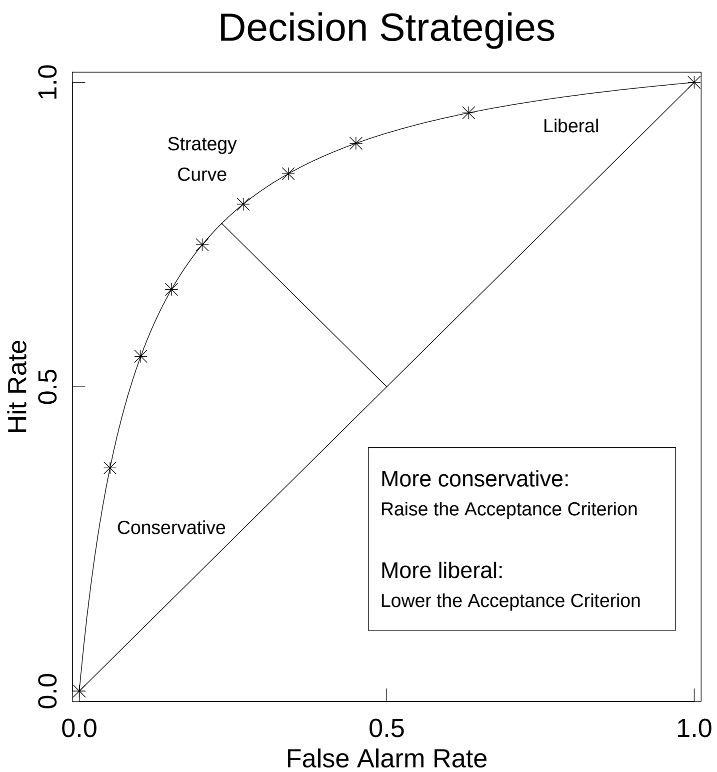


Figure 5: ROC Curve for Two-Choice Decision Environments (e.g. making Yes-No decisions for object classification and pattern recognition). ROC curves reveal trade-offs between error rates.

Each point on the ROC curve represents a particular decision strategy. It plots the relationship between the resulting Hit Rate and False Alarm Rate.

Finally, regardless of where our decision threshold is placed, the fundamental decidability of the decision task (or the detectability of the signal detection task) is measured by the quantity "d-prime" ($d'$). It is defined as the difference between the means of the two distributions, scaled by the square-root of their average variance (a conjoint standard deviation):

$$d' = \frac{|\mu_2 - \mu_1|}{\sqrt{\frac{1}{2}(\sigma_2^2 + \sigma_1^2)}}$$

where the two distributions are characterized by means $\mu_1$ and $\mu_2$ and standard deviations $\sigma_1$ and $\sigma_2$. An improvement in $d'$ can result either from pushing the two distributions further apart, or from making one or both of them narrower. In the ROC curve, $d'$ corresponds to how "bowed" the curve is. The bigger $d'$ is, the better; a pattern recognition problem with high decidability will have a large $d'$, so the curve approaches the upper-left corner. Any value higher than about 3 is great. The Figure below illustrates $d' = 11.36$ for iris recognition.

These considerations illustrate what might be called the "Primary Law of Pattern Recognition":

*The key issue is the relation between within-class variability and between-class variability.*

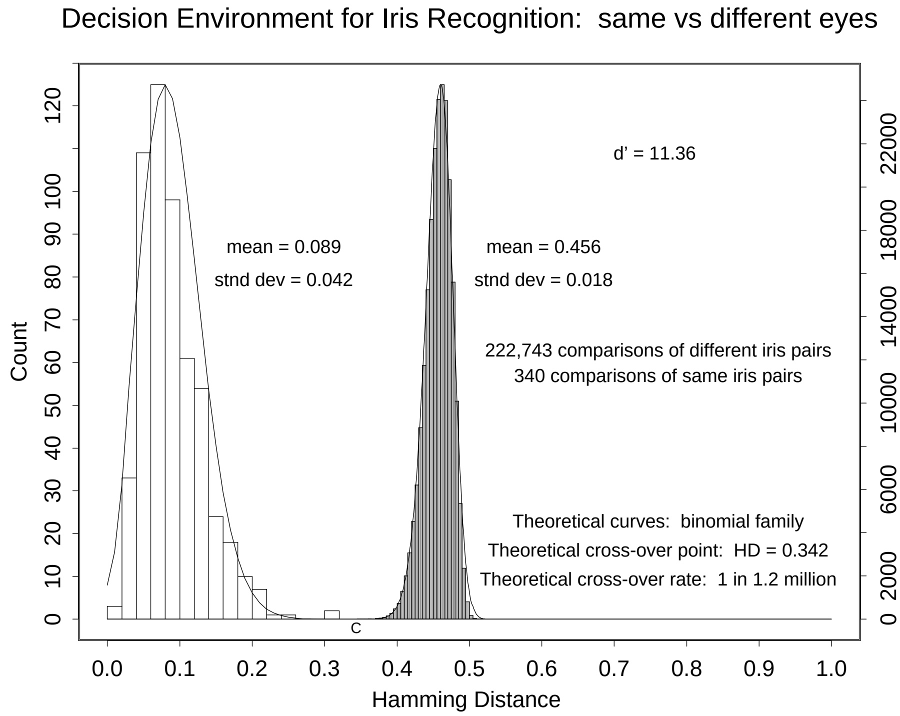


Figure 6: A powerful decision environment with $d' = 11.36$, illustrating how well-separated same/different distributions lead to highly decidable classification and pattern recognition.

## 11 Infering shape from shading: surface geometry.

As was illustrated earlier by the examples of infering surface and object properties from texture, colour, stereo, and motion information, the shading and brightness variation within an image is another important cue to surface shape.

As with all of these problems, computing "shape-from-shading" requires the disambiguation of many confounding factors. These arise from the

1. geometry of the illuminant (e.g. is the light a point source or extended? If a point source, where is it?) Are there several light sources? How will these affect the shading and shadowing information?
2. reflectance properties of the surface. What kind of surface is it – e.g. Lambertian, or specular, or a combination of both?
3. geometry of the surface (its underlying shape). Are shadows cast?
4. rotations of the surface relative to perspective angle and illuminant.
5. variations in material and surface reflectance properties across space (e.g. variation from Lambertian to specular where skin becomes more oily).
6. variations in surface albedo ("greyness")

## 12 Boundary descriptors; Fundamental Theorem of Curves; codons.

Closed boundary contours can be expanded into basis functions (such as "Fourier descriptors," or other functions) from their curvature map, in order to generate a shape description that is invariant to translation, rotation, and dilation. See next page (illustration of outline of a hand). By cataloging a list of all possible combinations of changes in sign of the curvature map relative to the zeroes of curvature, it is possible to generate a restricted "grammar" for the shapes of closed contours. A lexicon of all possible shapes having a certain number of zeroes-of-curvature generates a list of "codons," from which shapes can be classified and recognized. This approach is one of several proposing an elementary grammar for shape.

## 13 Object-centered coords. Solid parameterization. Superquadrics.

Represent solids by the unions and intersections of generalized superquadric objects, defined by equations of the form:

$$Ax^\alpha + By^\beta + Cz^\gamma = R \tag{19}$$

Examples include "generalized cylinders" and cubes (large exponents); prolate spheroids (footballs) and oblate spheroids (tomatoes), when $\alpha = \beta = \gamma = 2$ and when only two of $(A, B, C)$ are equal to each other.

These simple, parametric descriptions of solids, when augmented by Boolean relations for conjoining them, allows one to generate object-centered, "volumetric" descriptions of the objects in a scene (instead of an image-based description) by just giving a short list of 3D parameters and relations, rather like the codon descriptors for closed 2D shapes.

# 14 Inverse problems; energy minimization and regularization.

# 15 Model-based vision. Appearance-based vs volumetric models.

## 15.1 Deformable parametric models

A powerful approach for representing complex shapes in simple and compact (if only approximate) terms, is the use of *deformable parametric models.* These are especially useful for time-varying objects, such as a human face generating some expression that evolves in time, or a hand gesture. The idea is to find some model (such as the superquadrics) and a parameter set that are fitted to describe the object. They thereby constitute a compact code that can be used for detection and recognition of the object, although fitting such parameters to a 3D object is an "inverse problem" of high computational complexity. If these parameters are then made to evolve in time, as $(A(t), \alpha(t), \beta(t), ...)$ above, one can encode (compress, generate) an image sequence such as a "talking head" or avatar. This topic unites both computer vision and graphics. The new international MPEG-4 standard due next year for motion image encoding specifies provision for both facial animation and body animation.

# 16 Face recognition

*Louis Léopold Boilly,* **Reunion de Têtes Diverses**

The goal of detecting faces and recognizing their identity has long been one of the "Holy Grail" problems in computer vision. It is a hard problem for all of the reasons we have encountered that make computer vision hard:

- Faces are surfaces on 3D objects (heads). Therefore the images they project depend on the perspective angle between object and camera, the rotation of the object around its own axes, and the illuminant.
- Facial surfaces have relief, and so parts (e.g. noses) can occlude other parts. Hair can also create random occlusions and shadows.
- Surface relief causes shading and shadows to depend upon the angle of the illuminant, and whether it is an extended or a point source.
- Faces have variable specularity (dry skin may be Lambertian, oily or sweaty skin may be specular). As always, this confounds the interpretation of the reflectance map.
- Parts of faces can move around relative to other parts (eye movements; lip movements; eyebrows and winks).
- Humans put things on their faces (e.g. glasses, cosmetics, cigarettes) and change their facial hair (moustaches, eyebrows). They also use their faces as organs of expression, and so the surface isn't even rigid. (Ideally one would like not only to be able to detect and recognize faces, but also to interpret their expression!)

### 16.1 Issues

As usual, the task of face recognition by computer raises questions such as

1. What is the best representation to use?
2. Must this be treated as a 3D (object-based) or 2D (image-based) problem?
3. How can invariances to size (hence distance), location, tilt, and angle of view be achieved? (A face should acquire the same representation under such transformations, for matching purposes.)
4. What are the *generic* (i.e. universal) properties of all faces that we can rely upon for the detection problem?
5. What are the *particular* features that we can rely upon to recognize the identity of any given face?
6. What is the best way to handle "integration of evidence" and make decisions under uncertainty?

### 16.2 History

Early attempts were based on simple "template-matching" on a pixel-by-pixel basis. Although primitive, these methods have now re-surfaced as performing about as well as anything else. (e.g. Brunelli and Poggio, 1995). Many methods sought to enumerate a long vector of facial features, such as measurements of the distances between many points on the face. These include all mutual distances between distinct points such as the pupils, the boundary of the chin, the cantis (corners) of the eyes, corners of the lips, etc. However, these methods were generally just not very robust, despite the combinatorial explosion of possible measurements and metrics.

The most prominent method in the last decade has been the *Eigenfaces* approach (Kirby and Sirovich; Turk and Pentland): Perform a complete Karhunen-Loeve Transform of a large database of faces (e.g. 8,000) to extract the principle components, i.e. the 20 or so main factors along which different faces differ from each other. These may be expressed as the eigenvalues on the eigenvectors (eigenfaces), which form a new abstract kind of feature vector on which to base recognition. Performance is usually in the 90% to 95% range. However, a limitation in this methods is that many of the principle components just extract variations due to shading caused by variations in the angle of illumination! Thus the method is essentially a 2D representation of faces, and lacks invariances to illumination.

### 16.3 The Fundamental Problem.

The central issue in pattern recognition is the relation between within-class variability and between-class variability. These are determined by the degrees of freedom spanned by the pattern classes. Ideally the within-class variability should be small and the between-class variability large, so that the classes are well separated. In the case of encoding faces for identity, one would like different faces to generate face codes that are as different from each other as possible, while different images of the same face should ideally generate similar codes across conditions. Several recent investigations of how well this goal is achieved have studied the invariances in face coding schemes under changes in illumination, perspective angle or pose, and expression. Their results have tended to show that there is greater variability in the code for a given face across these three types of changes, than there is among the codes for different faces when these three factors are kept constant [e.g. Adini et al., 1997]. Since reports documenting performance of particular face recognition algorithms have often been based upon trials in which these factors (pose, illumination, and expression) were held artificially constant, the performance statistics in real-world settings have been disappointing by contrast.

When there is variability across two or more dimensions (let us say both face identity and facial expression), then discriminability can benefit from variability *within* a class of the other dimension, but not *between* classes of the other dimension. For example, facial expressions are more reliably distinguished if there is large variation among the different expressions generated by a given face, but small variation in how a given expression is generated among different faces. Consequences of within-class and between-class variability, for single dimensions and across them, are noted in the following table:

| **Task** | **Within-Class Variability** | **Between-Class Variability** |
|---|---|---|
| **Face detection** (classes: face / non-face) | bad | good |
| **Face identification** (classes: same/different faces) | bad | good |
| **Facial expression interpretation** (classes: same/different faces) | good | bad |
| **Facial expression interpretation** (classes: same/different expressions) | bad | good |

Some of the forms of variation in facial appearance that have been briefly mentioned here were perhaps best summarized pictorially in the painting by Boilly, *Reunion de Têtes Diverses,* shown earlier. More systematic illustrations of these individual forms of variation (such as lighting, or identity, while holding constant the other ones such as expression) are shown in these collages.

In characterizing the within-class variability and the between-class variability of faces, it is clear that over sufficient periods of time, the variability of any given face can easily outstrip the variability among contemporary faces. No one would deny that young babies look far more similar to each other, than each does to the adult that it grows into.

Even when all other factors such as pose angle, expression, illumination, and age are held constant, we can distinguish those aspects of facial variation that are genetically inherited ("genotypic features"), from those that primarily reflect development, aging, or environment ("phenotypic features"). Persons who are genetically identical would share all their genotypic features, such as gender, blood group, race, and DNA sequence, whereas phenotypic features can be shared among different individuals only by chance, according to their associated probability distributions.

One source of evidence about the genotypic/phenotypic ratio of facial variation arises from identical (monozygotic) twins. Obviously any pair of twins are always matched in age. Each twin's appearance changes over time in the normal dramatic way, yet the pair usually remain strikingly similar to each other in appearance at any age. Nobody would deny that identical twins look vastly more similar to each other than unrelated persons do. Since such twins are genetically identical, their similarity in appearance serves to calibrate the extent of *genetic penetrance* for facial structure.

A further, but secondary, calibration for the genetic penetrance of facial appearance is provided by persons who share only 50% rather than 100% of their genes. These include fraternal twins, full siblings, double cousins, and a given parent and offspring. Occasionally the latter pairings have virtually indistinguishable appearance at a similar age, such as Robert F. Kennedy and his son Michael in adulthood.

### 16.4 Genotype-Phenotype Limitations on Face Recognition

Automatic face recognition systems have two basic error rates: a False Accept Rate (accepting one person as another), and a False Reject Rate (failing to detect a valid match). It is clear that the phenotypic variation of a person's facial appearance over time imposes one limit on performance (not necessarily the lowest limit), since when such variation is great enough, it causes a False Reject. Likewise, it is clear that the high genetic penetrance for facial appearance imposes a different limit on performance (again, not necessarily the lowest such limit), since persons with identical genetic constitution look so similar. Thus the birth rate of identical twins determines one bound on the False Accept Rate, unless it can be demonstrated that face recognition algorithms can reliably distinguish identical twins, which has not yet been done. These two performance bounds are summarized in the following table:

| **Type of Feature** | **Performance Limitation** |
|---|---|
| Genotypic | False Accept Rate $\geq$ birth rate of monozygotic twins |
| Phenotypic | False Reject Rate $\geq$ feature variability over time |

Roughly one in 80 births are twins, and about a third of these are identical (monozygotic). So a representative sample of 240 births (counting twin births as one event) usually yields 243 persons, among whom there are three pairs of twins; and one such pair of persons are genetically identical. Thus the chances are roughly one in 121 (or $2/243 = 0.82\%$) that any person selected at random has an identical twin. Unless and until face recognition algorithms can demonstrate reliable distinction of twins, it must be assumed that such persons would be confused with each other, creating a minimum False Accept Rate of 0.82% due to this birth rate alone. For face recognition applications that require higher confidence levels than 99.18% for correct rejection of false matches, this ceiling on performance is a fatal limitation. Exactly the same argument, and numbers, pertains to the maximum possible confidence levels that can theoretically be achieved by DNA tests.

In summary, performance of face recognition schemes is bounded at one limit by a *genotypic* error rate (the birth rate of identical twins), and at another limit by a *phenotypic* error rate (change in facial appearance over time). These set minimal False Accept and False Reject frequencies, by undermining the between-class variability in the first case, and increasing the within-class variability in the second. It would be preferable to base recognition decisions upon features which had very little genetic penetrance, yet high complexity, and stability over the lifetime of the individual. Phenotypic facial features do exist with exactly these properties.

It is possible to base practical face recognition systems upon phenotypic facial information (not shared by identical twins) that possesses great complexity and randomness; and which furthermore shows invariance to imaging conditions, age, and expression. When such images can be acquired, they permit a massive reduction in both the False Accept and False Reject error rates compared with the genotypic error bound and phenotypic error bound noted in the above table for overall face recognition.

### 16.5 Phenotypic Facial Features with Great Complexity

By far the most numerous and dense degrees-of-freedom (forms of variability among individuals), which are both stable over time and visible in a face, are found in the complex texture of the iris of either eye. This protected internal organ, which can be imaged adequately at distances of up to about a meter, reveals about 266 independent degrees-of-freedom of textural variation across individuals. One way to calibrate the "information density" of the iris is by its human-population entropy per unit area. This works out to 3.4 bits per square millimeter on the iris, based upon 222,743 IrisCode comparisons recently collected by the British Telecom Research Laboratories. Iris encoding is based upon "texture demodulation" using the complex-valued 2D Gabor wavelets shown earlier, and statistical decision theory outlined in section 10.3.

### 16.6 Wavelet Approaches to Global Face Recognition

Recently much interest has developed in *wavelet* representations of entire faces. This idea can be applied in either a 2D or a 3D fashion, either to represent image structure or to represent the surface of the face as a 3D model. Because wavelets are localized, they can track changes in facial expression in a local way. Because wavelets are *scale specific*, they can serve both for the global (gross) task of "detecting" faces generically, and for the more detailed task of identifying those particular properties unique to a given face.

Thus this approach essentially treats a face as a kind of *texture*, made up of various undulations in various positions, sizes, and orientations. Remarkably, the major facial features such as eyes, lips, and noses can be extremely well represented by just a handful of 2D Gabor wavelets, as was illustrated on pages 24 and 26.

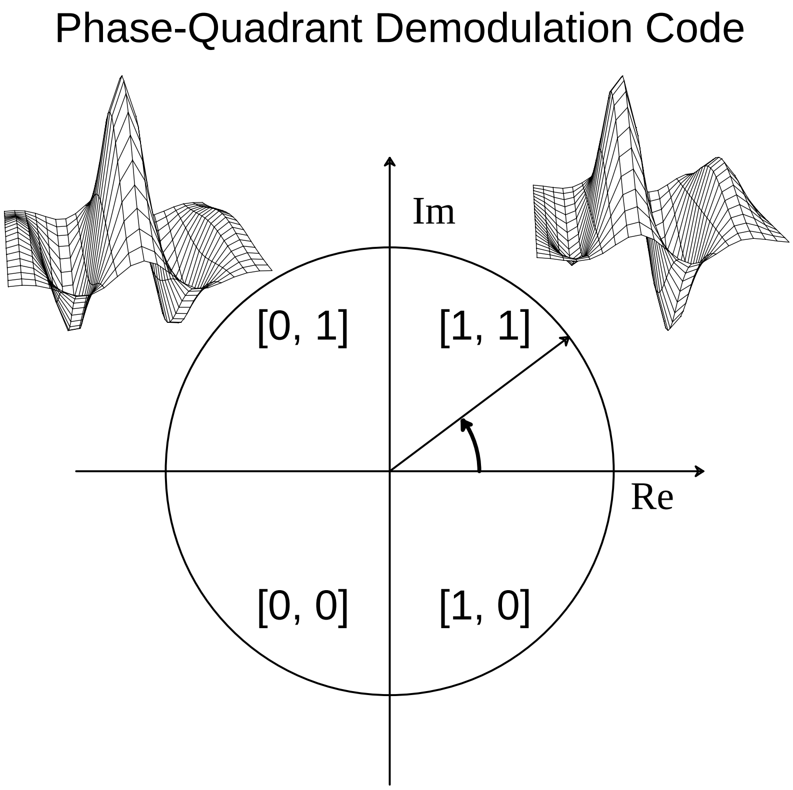


Figure 7: An encoding approach for faces (and other patterns) by phase-quadrant quantization of complex-valued 2D Gabor wavelet projections of image structure. Bits in the "face code" are set by the quadrant in which the phasor lies for each aspect of facial structure.