Computer Vision

Computer Science Tripos: 12 Lectures by C P Town

1. Overview. Goals of computer vision; why they are so difficult.
2. Image sensing, pixel arrays, cameras. Image coding. Biological vision.
3. Mathematical operations for extracting structure from images.
4. Edge detection operators; the Laplacian and its zero-crossings.
5. Multi-scale feature detection and matching.
6. Texture, colour, stereo, and motion descriptors. Disambiguation.
7. Lambertian and specular surface properties. Reflectance maps.
8. Shape description. Codons; superquadrics and surface geometry.
9. Perceptual psychology and visual cognition. Visual illusions.
10. Bayesian inference. Classifiers; probabilistic methods.
11. Learning and statistical methods in vision. Optical character recognition and Content based image retrieval.
12. Face detection, face recognition, and facial interpretation.

Syllabus

Aims

The aims of this course are to introduce the principles, models and applications of computer vision. The course will cover: image formation, structure, and coding; edge and feature detection; texture, colour, stereo, and motion; wavelet methods for visual coding and analysis; interpretation of surfaces, solids, and shapes; appearance modelling; pattern recognition and classification; visual inference and learning. Several of these issues will be illustrated using the examples of optical character recognition, image retrieval, and face recognition.

Lectures

- **Goals of computer vision; why they are so difficult.** How images are formed, and the ill-posed problem of making 3D inferences from them about objects and their properties.
- **Image sensing, pixel arrays, cameras.** Elementary operations on image arrays; coding and information measures. Sampling and aliasing. Biological vision.
- **Mathematical operators for extracting image structure.** Finite differences and directional derivatives. Filters; convolution; correlation. Fourier and wavelet transforms.
- **Edge detection operators; the information revealed by edges.** The Laplacian operator and its zero-crossings. Logan's theorem.
- **Multi-scale feature detection and matching.** Gaussian pyramids and SIFT (scale-invariant feature transform). Active contours; energy-minimising snakes. 2D wavelets as visual primitives.
- **Texture, colour, stereo, and motion descriptors.** Disambiguation and the achievement of invariances. Image and motion segmentation.
- **Lambertian and specular surfaces.** Reflectance maps. Image formation geometry. Discounting the illuminant when inferring 3D structure and surface properties.
- **Shape representation.** Inferring 3D shape from shading; surface geometry. Boundary descriptors; codons; superquadrics and the "2.5-Dimensional" sketch.
- **Perceptual psychology and visual cognition.** Vision as model-building and graphics in the brain. Learning to see. Visual illusions, and what they may imply about how vision works.
- **Bayesian inference in vision; knowledge-driven interpretations.** Classifiers and pattern recognition. Probabilistic methods in vision.
- **Applications of machine learning in computer vision.** Discriminative and generative methods. Optical character recognition. Content based image retrieval.
- **Approaches to face detection, face recognition, and facial interpretation.** Appearance and model based representations. 2D and 3D approaches. Cascaded detectors.

Objectives

At the end of the course students should:

- understand visual processing from both "bottom-up" (data oriented) and "top-down" (goals oriented) perspectives;
- be able to decompose visual tasks into sequences of image analysis operations, representations, specific algorithms, and inference principles;
- understand the roles of image transformations and their invariances in pattern recognition and classification;
- be able to describe and contrast techniques for extracting and representing features, edges, shapes, and textures
- be able to analyse the robustness, brittleness, generalizability, and performance of different approaches in computer vision;
- understand some of the major practical application problems, such as face interpretation, character recognition, and image retrieval.

Recommended books

* Forsyth, D. A. & Ponce, J. (2003). *Computer Vision: A Modern Approach.* Prentice Hall.

Shapiro, L. & Stockman, G. (2001). *Computer vision.* Prentice Hall.

Online resources

The OpenCV Computer Vision Library: [an excellent C++ open source library with interfaces for some other languages]
http://opencv.willowgarage.com

"CVonline: The Evolving, Distributed, Non-Proprietary, On-Line Compendium of Computer Vision" (Edinburgh University):
http://homepages.inf.ed.ac.uk/rbf/CVonline/

Matlab Functions for Computer Vision and Image Processing:
http://www.csse.uwa.edu.au/∼pk/Research/MatlabFns

Annotated Computer Vision Bibliography:
http://iris.usc.edu/Vision-Notes/bibliography/contents.html

"The Computer Vision Homepage" (Carnegie Mellon University): [somewhat outdated]
http://www-2.cs.cmu.edu/∼cil/vision.html

Exercises

This lecture course and the accompanying notes are largely based on the Computer Vision course taught by Professor John Daugman in previous years. However, the syllabus and notes have been revised this year, and as a consequence a few of the exam questions set in previous years are no longer relevant.
A collection of exercises for this course, some of them based on relevant past Tripos Questions and including Model Answers, is provided on the course website:
http://www.cl.cam.ac.uk/teaching/1011/CompVision

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 cameras.

Some examples of computer vision applications and goals:

- automatic face recognition, and interpretation of expression
- visual guidance of autonomous vehicles
- automated medical image analysis, interpretation, and diagnosis
- robotic manufacturing: manipulation, grading, and assembly of parts
- OCR (optical character recognition): recognition of printed or handwritten characters and words
- ANPR: automated number plate recognition
- agricultural robots: visual grading and harvesting of produce
- smart offices: tracking of persons and objects; understanding gestures
- biometric-based visual identification of persons
- security monitoring and alerting; detection of anomaly
- 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
- image search and matching by content

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 video camera, 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!

   A classical and central problem in computer vision is face recognition. Humans perform this task effortlessly, rapidly, reliably, and unconsciously. (We don't even know quite how we do it; like so many tasks for which our neural resources are so formidable, we have little "cognitive penetrance" or understanding of how we actually perform face recognition.) Consider these three facial images (from Pawan Sinha, MIT, 2002):

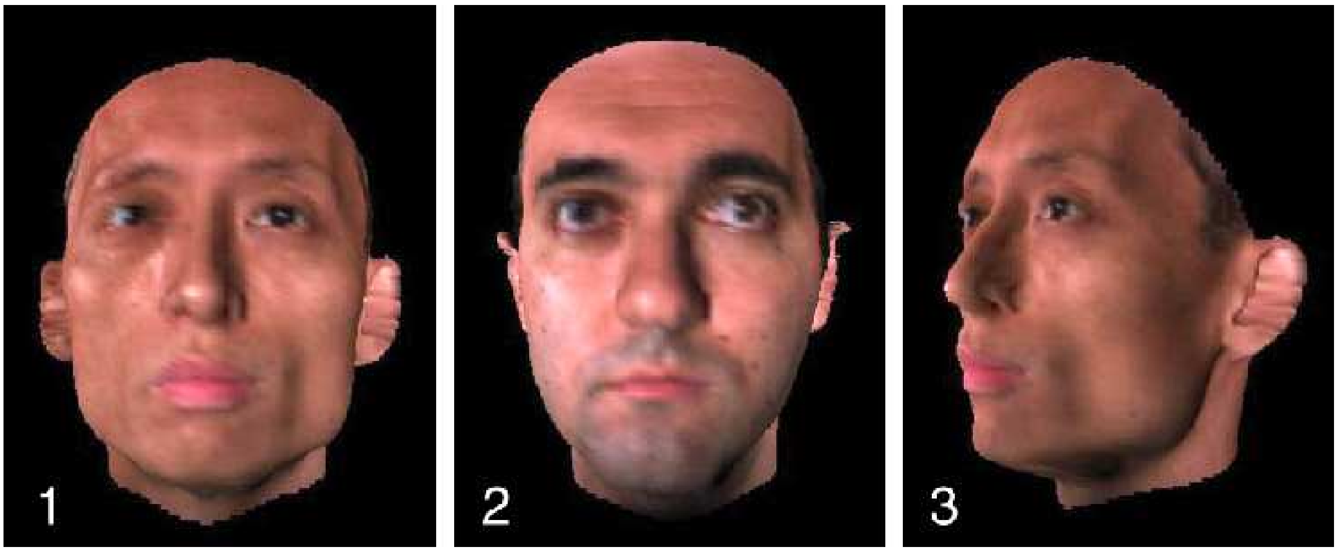
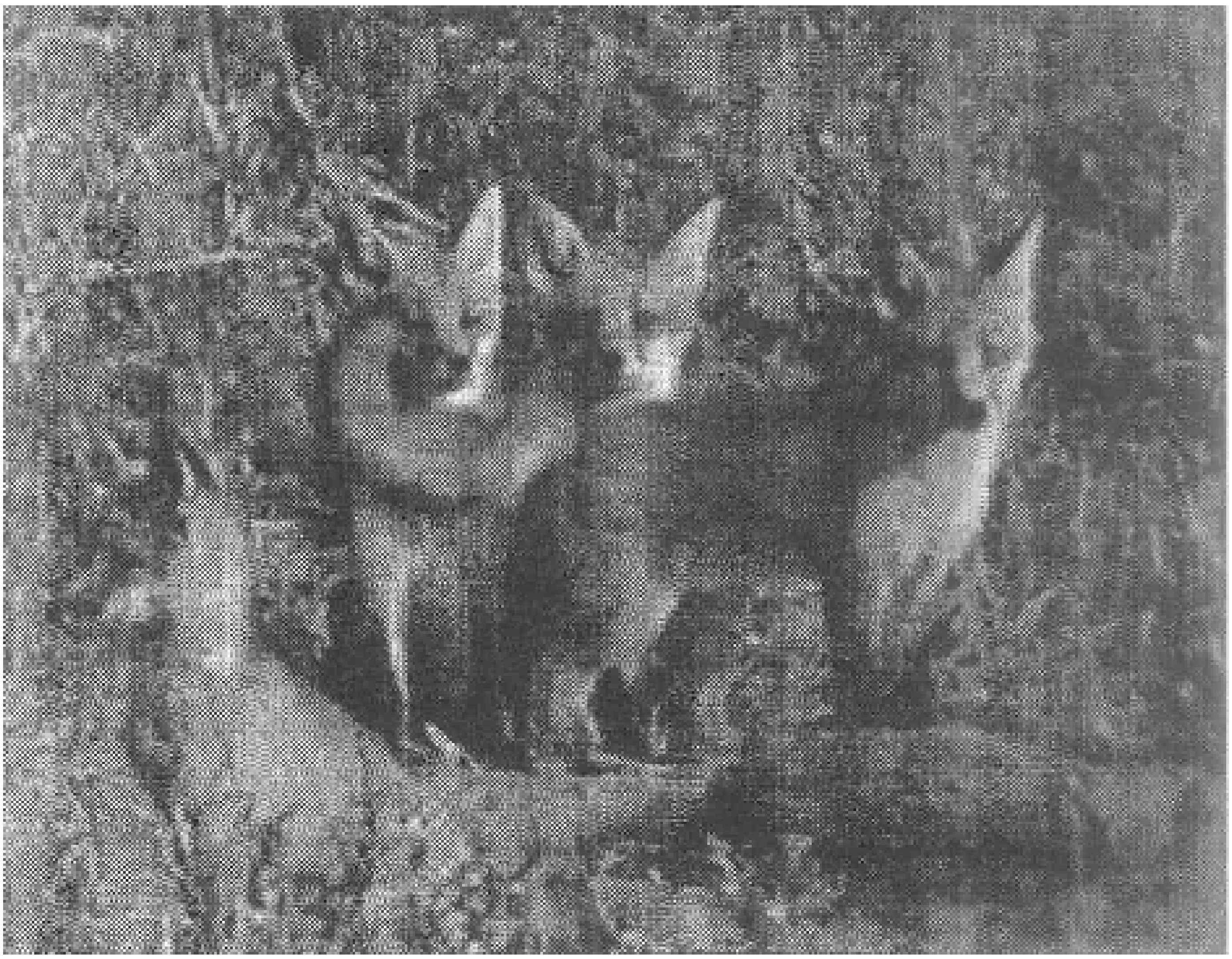


Figure 1

Which two pictures show the same person?

Most algorithms for computer vision select 1 and 2 as the same person, since those images are more similar than 1 and 3.

2. Very few visual tasks can be successfully performed in a purely data-driven way ("bottom-up" image analysis). Consider the next image example: 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



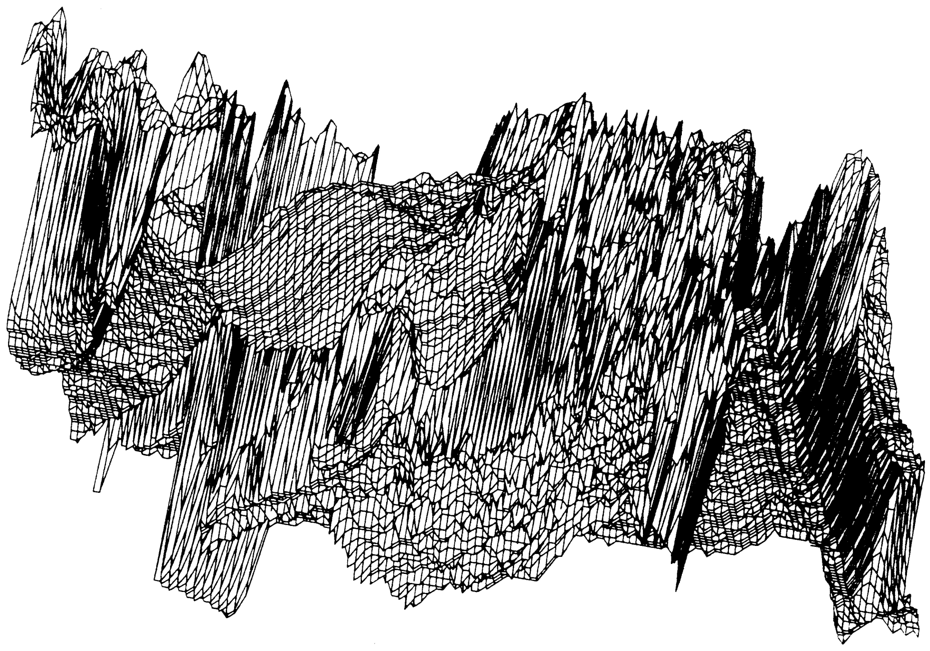
an image that can:

- 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.)

Here is a video demo showing that computer vision algorithms can infer 3D world models from mere 2D (single) images, and navigate within them: http://www.youtube.com/watch?v=VuoljANz4EA .

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:

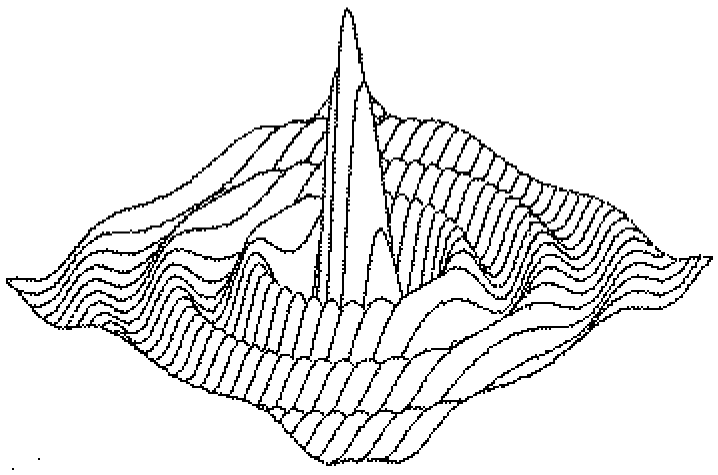
Most of the problems we need to solve in vision are *ill-posed,* in Hadamard's sense that a *well-posed* problem must have the following set of properties:

- 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:

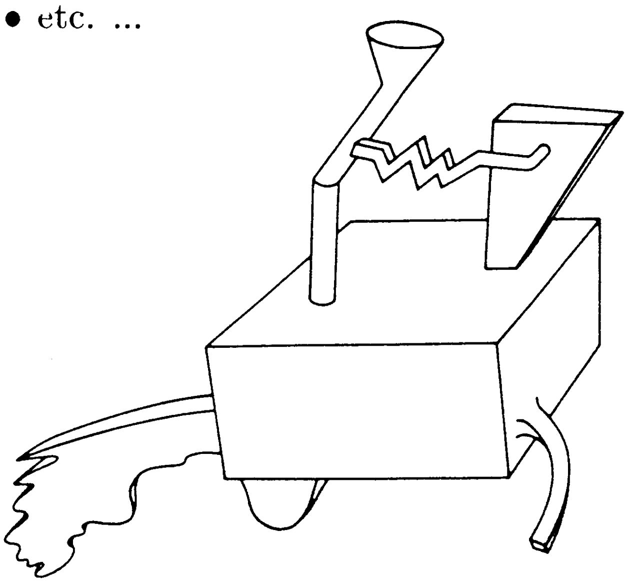
- inferring depth properties from an image
- inferring surface properties from image properties

- inferring colours in an illuminant-invariant manner
- inferring structure from motion, shading, texture, shadows, ...
- inferring a 3D shape unambiguously from a 2D line drawing:



- interpreting the mutual occlusions of objects, and stereo disparity
- recognising 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 how images can be acquired and represented.

2 Image sensing, pixel arrays, CCD cameras, image coding.

A CCD 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) ranges from a few 100,000 to many millions (*e.g.* 10 MegaPixel) in an imaging array that is about 1 cm^2 in size, so each pixel sensing element is only about 3 microns in width. The photon flux into such small catchment areas is a factor limiting further increases in resolution by simply building denser imaging arrays. Note also that 3 microns is only 6 times larger than the wavelength of a photon of light in the visible spectrum (yellow $\sim$ 500 nanometers or nm).

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 digitiser, and by the inherent signal-to-noise ratio of the CCD array.

Digital cameras have far surpassed film in terms of sensitivity to light, with ISO equivalent speeds of up to 102,400, a number that was previously unimaginable using conventional film photography.
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. In the case of composite (analog) video, colour is encoded either as a high-frequency "chrominance burst" (to be separately demodulated and decoded); or else put on a separate channel ("luma" and "chroma" portions of an S signal); or else provided as three separate RGB colour channels (red, green, blue). Colour information requires much less resolution than luminance, and some coding schemes exploit this.

A framegrabber or a strobed sampling block in a digital camera contains a high-speed analogue-to-digital converter which discretises this video signal into a byte stream. Conventional video formats include NTSC (North American standard): 640×480 pixels, at 30 frames/second (actually there is an interlace of alternate lines scanned out at 60 "fields" per second); and PAL (European, UK standard): 768×576 pixels, at 25 frames/second.

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

2.1 Image formats and sampling theory

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. An image with an oblique edge might look like:

0	0	0	1	1	0
0	0	1	2	10	0
0	1	2	17	23	5
0	3	36	70	50	10
1	10	50	90	47	12
17	23	80	98	85	30

There are many different image formats used for storing and transmitting images in compressed form, since raw images are large data structures that contain much redundancy (e.g. correlations between nearby pixels) and thus are highly compressible. Different formats are specialised for compressibility, manipulability, or the properties of printers and browsers. Some examples:

- `.jpeg` - ideal for variable "lossy" compression of continuous colour images, with a "quality factor" (typically 75) that can be specified. Useful range of DCT compression goes from 100:1 ("lossy") to about 10:1 (almost lossless).
- `.mpeg` - a stream-oriented, compressive encoding scheme used mainly for video (but also multimedia). Individual image frames are `.jpeg` compressed, but an equal amount of redundancy is removed temporally by inter-frame predictive coding and interpolation.
- `.gif` or `.png` - Uses lossless data compression and is favoured for websites. GIF is limited to 8 bits of colour information, PNG also supports 24-bit RGB.
- `.tiff` - A complex umbrella class of tagged image file formats with randomly embedded tags and up to 24-bit colour. Non-compressive (although some variants support zip-like compaction).
- `.bmp` - a non-compressive bit-mapped format in which individual pixel values can easily be extracted.

In addition there are varieties of colour coordinates used for "colour separation," such as HSI (Hue, Saturation, Intensity), or RGB (Red, Green, Blue), CMY, etc. But regardless of the sensor properties and coding format used, ultimately the image data must be represented numerically pixel by pixel.

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.
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 quantised 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 quantised 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 normalised) *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.

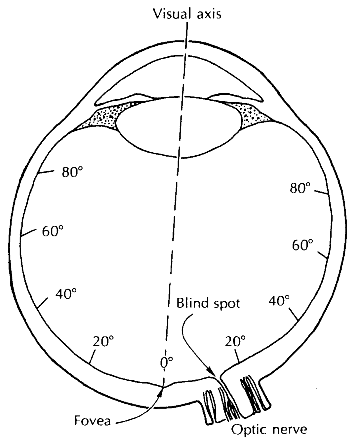
The discreteness of image arrays imposes an upper limit on the amount of information they can contain. One way to describe this is by the total bit count, but this does not relate to the optical properties of image information. A better way is through *Nyquist's Theorem*, which tells us that the highest *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.

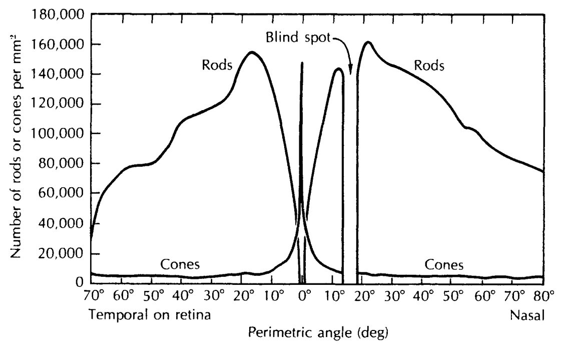
2.2 Biological image acquisition: early vision from retina to primary cortex.

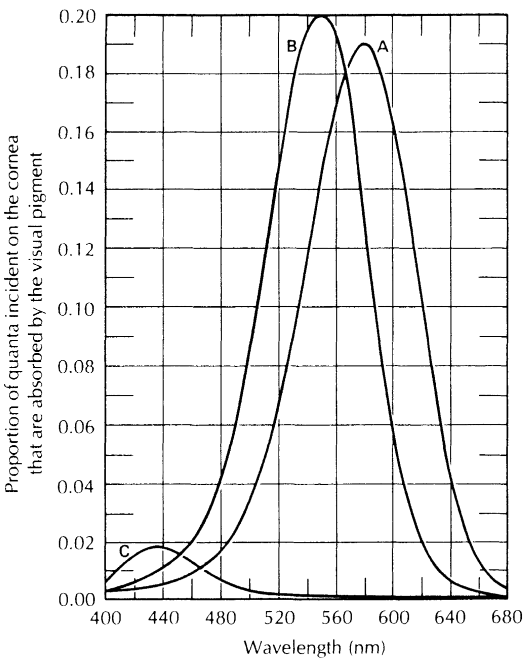
A strategy that has long inspired researchers in Computer Vision, whether they work on low-level problems (such as sensor design, image coding, and feature extraction), or high-level problems (such as pattern recognition, inference, and visual learning), is:

$$\boxed{\textbf{Neurobiological Visual Principles} \Longrightarrow \textbf{Machine Vision}}$$

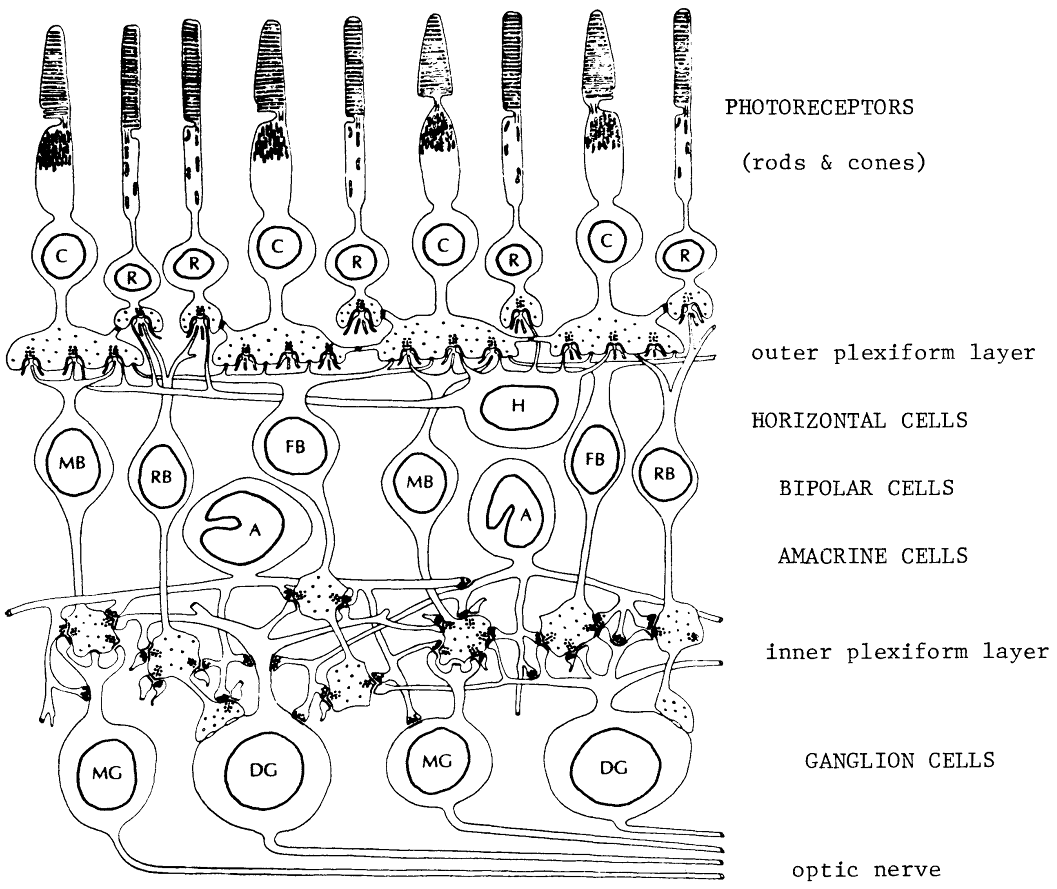
The structure of biological nervous tissue and the nature of events that occur in it are utterly different from those found in computing hardware. Yet since the only general-purpose visual systems that exist today are the biological ones, let us learn what we can from "wetware."
Overall, the human brain contains about 100 billion neurons (10^{11}). On average each neuron may have connections with about 1,000 to 10,000 others, and so the total number of synapses (= junctions between neurons) in the brain is a staggering 10^{15}. Yet balanced against this massive connectivity is the surprising sluggishness of neurons: the time course of nerve impulse generation prevents "clocking" of nerve pulses any faster than about 300 Hz. Neural activity is fundamentally <u>asynchronous</u>: there is no master clock on whose edges the events occur. A further contrast with computing systems is that it is rarely possible to distinguish between <u>processing</u> and <u>communications</u>, as we do in computing. In the brain, there are just impulses implementing both, by exchange of signals amongst neurons. It is not so much a hierarchical architecture as a <u>parallel</u> one, with reciprocal connections amongst different areas. About $2/3^{rds}$ of the brain receives visual input; we are quite fundamentally visual creatures. There are some 30 known different visual areas, of which the <u>primary visual cortex</u> in the occipital lobe at the back of the brain has been the most extensively studied.







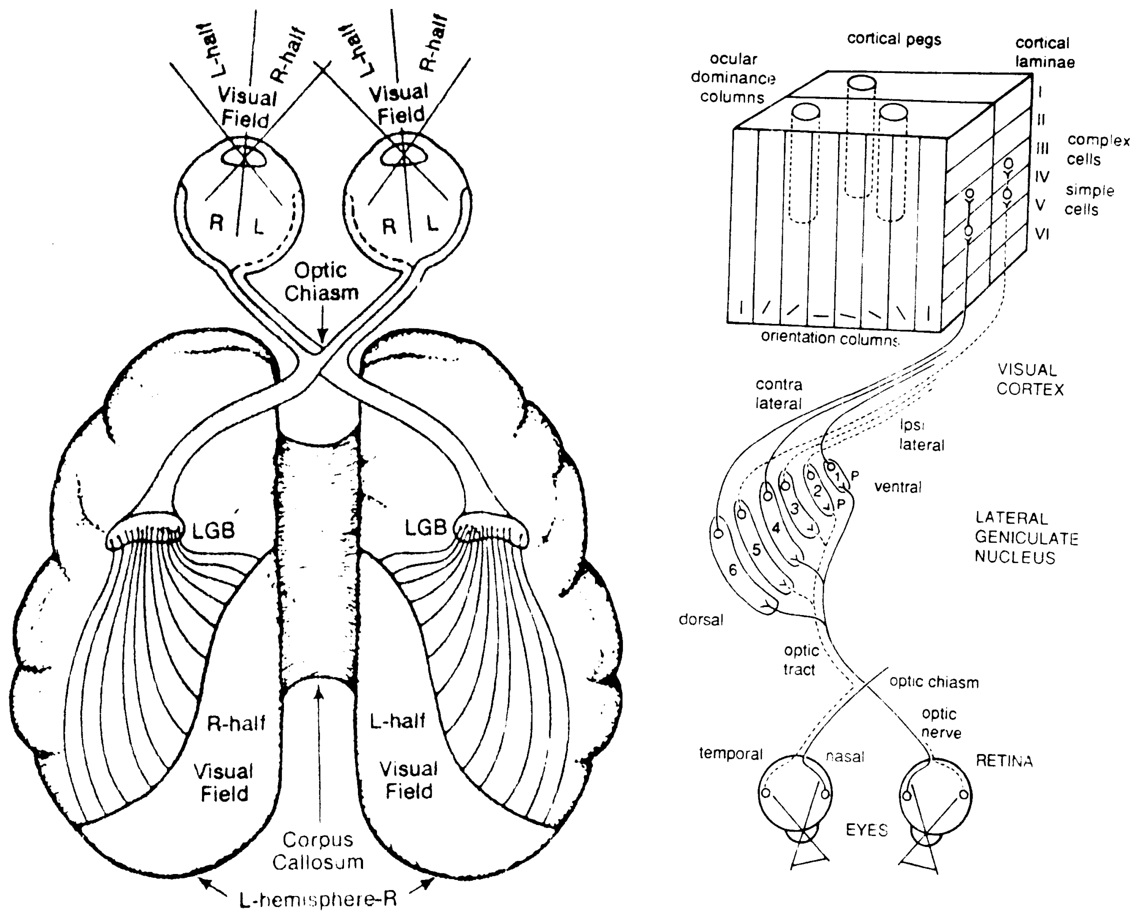
The mammalian eye is formed from a collapsed ventricle of the brain. The retina is about 1 mm thick and contains about 120 million light-sensitive photoreceptors, of which only 6 million are cones (in 3 wavelength-selective classes nominally red, blue, and green) and the vast remainder are rods which do not discriminate in wavelength. The visible spectrum of light consists of wavelengths in the range of 400nm - 700nm. Rods are specialised for much lower light intensities than cones; they subserve our "night vision" (hence the absence of perceived colour at night), and they pool together their responses (hence their much poorer spatial resolution). Cones exist primarily near the fovea, in about the central 20° (see diagram), where their responses remain individual and thus they detect with high spatial resolution. But cone light sensitivity is much less than rods, functioning only at higher light levels, and so we really have a dual system with two barely overlapping dynamic ranges. The total dynamic range of human vision (range of light intensities that can be processed) is a staggering 10^{11} to 1. At the low end, we can reliably "see" individual photons (i.e. reliably have a visual sensation when at most a few photons reach the retina in a burst).



The retina is a multi-layered structure, containing 3 nuclear layers (of neurons) plus 2 plexiform layers (for interconnections amongst the neurons). Paradoxically, the photoreceptors are at the back, so light must first travel through all of the rest of the retina before being absorbed by the pigments in the rods and cones. Only in a very crude sense can one describe the retina as an "image capture" device like a camera, having analogue input phototransducers that convert photons into voltage changes, and discrete output devices that send pulses down the optic nerve. This simple view is quickly discarded by recognising that there are 120 million "input channels" (the photoreceptors, similar in a sense to pixels), but only 1 million "output channels" (the axons of the ganglion cells which constitute the optic nerve). Clearly the retina is already doing a lot of processing of the image, and it sends its coded results to the brain: not merely a raw converted image array. The retina *is a part* of the brain.

2.3 Receptive field structure in the retina

The spatial structuring of excitatory and inhibitory influences amongst neurons in the retina gives them their properties as image operators. Similarly for the temporal structure of their interactions. In both space and time, retinal neurons can thus be described as filters; and to the extent that they act as



Schematic of the central visual pathway in the human. (from Popper and Eccles, 1977)

linear devices (having the properties of proportionality and superposition of responses to components of stimuli), their behaviour can be fully understood (and even predicted for arbitrary images) through Fourier analysis and the other tools of linear systems analysis. An important aspect of retinal receptive fields – as distinct from those found in most neurons of the visual cortex – is that their spatial structure is isotropic, or circularly symmetric, rather than oriented.

2.4 Visual cortical architecture and receptive field structure

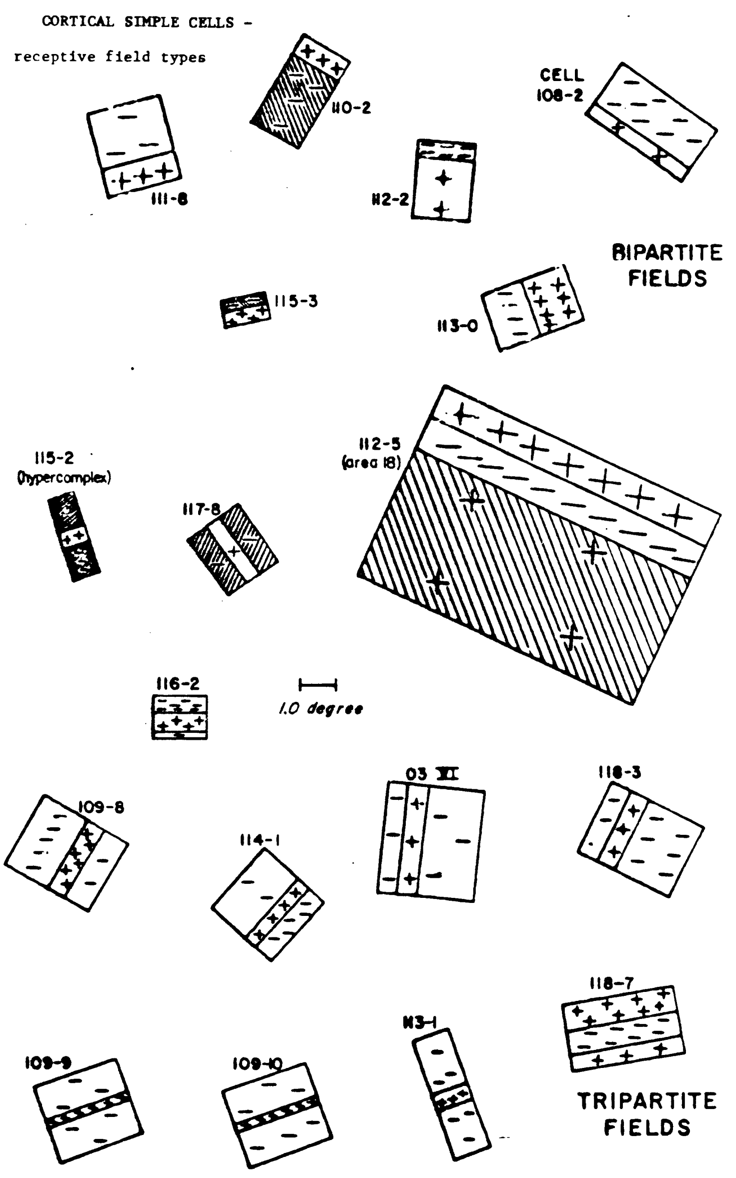
The optic nerve from each eye splits into two halves at the optic chiasm, each portion continuing on to only one of the two cerebral hemispheres of the brain. The optic nerve portion containing signals from the nasal half of each retina crosses over to project only to the contralateral (opposite) brain hemisphere; whereas the optic nerve portion bearing signals from the temporal half of each eye projects only to the ipsilateral (same side) brain hemisphere. Since the optical image on each retina is inverted, this means that the left-half of the visual world (relative to the point of gaze fixation) is directly "seen" only by the right brain; and the right-half of the visual world only by the left brain.

The optic nerve projections to each visual cortex pass first to a 6-layered structure called the lateral geniculate nucleus (LGN), in a polysensory organ of the midbrain called the thalamus. It is an intriguing fact that this so-called "relay station" actually receives 3 times more descending (efferent) fibres projecting back down from the cortex, as it does ascending (afferent) fibres from the eyes. Could it be that this confluence compares cortical feedback representing hypotheses about the visual scene, with the incoming retinal data in a kind of predictive coding or hypothesis testing operation? Several scientists have proposed that "vision is graphics" (i.e. what we see is really our own internally generated 3D graphics, modelled to fit the 2D retinal data, with the model testing and updating occurring here in the thalamus).

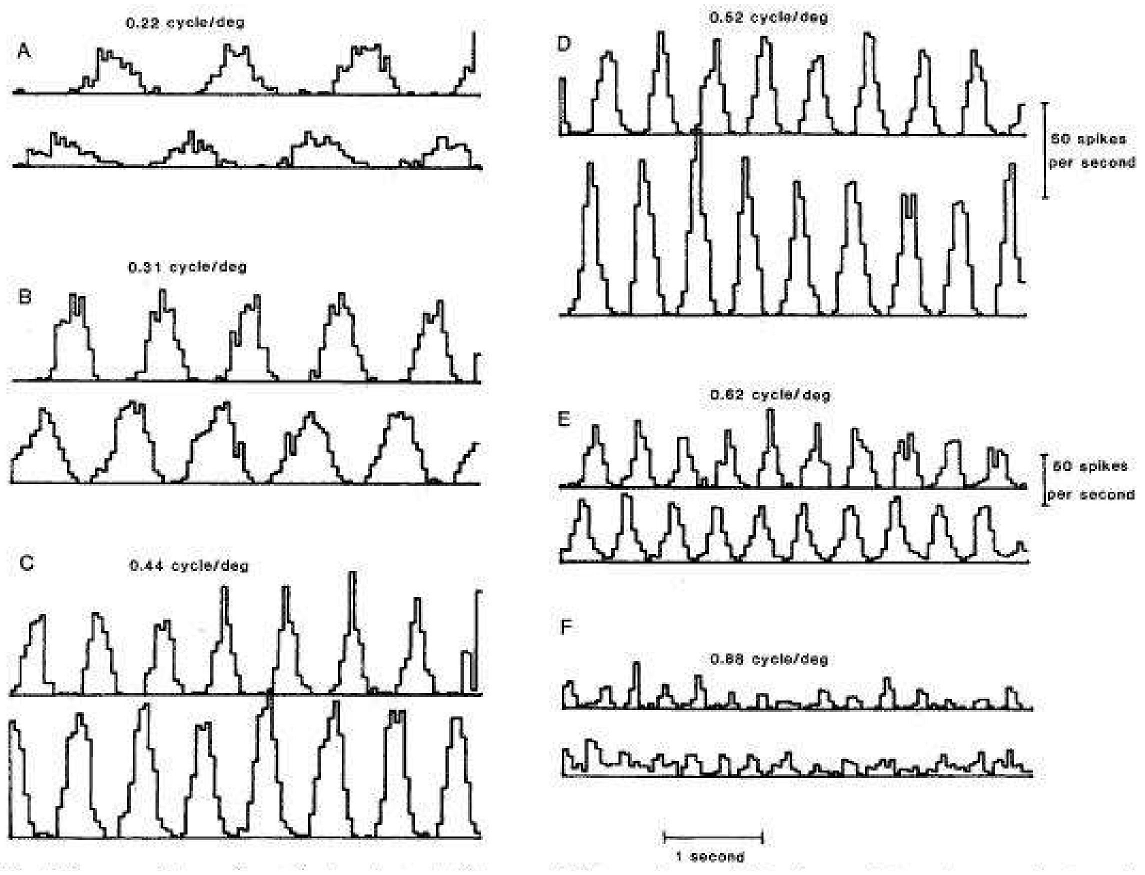
The right-eye and left-eye innervations from each LGN to the primary visual cortex in the occipital lobe of that hemisphere are inter-woven into "slabs," or columns, in which neurons receive input primarily from just one of the eyes. These ocular dominance columns alternate with a cycle of about 1 mm. Clearly each hemisphere is trying to integrate together the signals from the two eyes in a way suitable for stereoscopic vision, by computing the relative retinal disparities of corresponding points in the two images. The disparities reflect the relative positions of the points in depth, as we will study later with stereoscopic visual algorithms.

When individual neurons in the visual cortex are probed with microelectrodes during light stimulation of the retina, their functional properties are revealed by demarcating the region of visual space over which they respond (as indicated by a change in their firing rate). Areas where they are excited by light are indicated by + marks; areas where light inhibits them are indicated by − marks. Their plotted receptive fields then seem to reveal 5 main spatial "degrees of freedom:"

1. Position of their receptive field in visual space, both horizontally...
2. ...and vertically;
3. Size of their receptive field;
4. Orientation of the boundaries between excitatory and inhibitory regions;
5. Phase, or symmetry of the receptive field (bipartite or tripartite types).

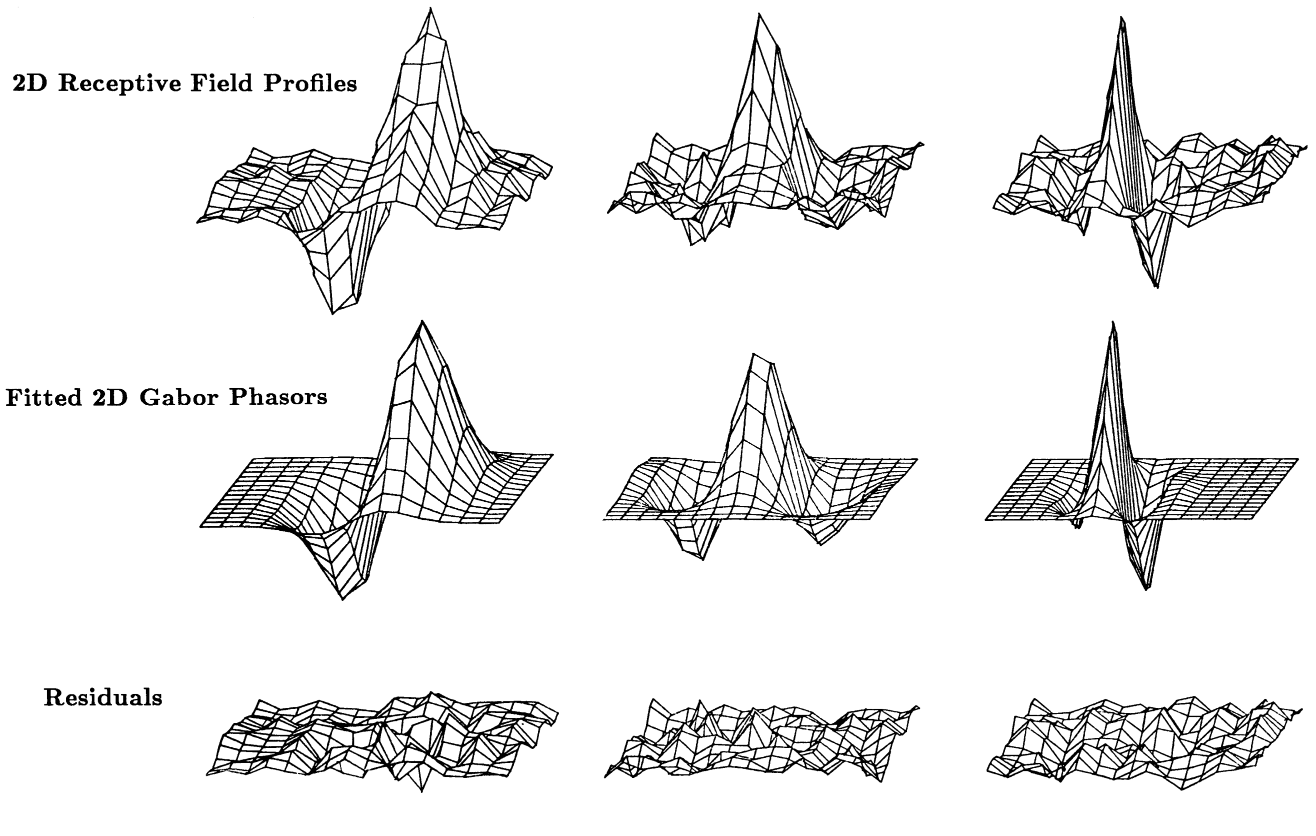


Finally, by plotting the actual amount by which a neuron is excited or inhibited by light, as a function of the coordinates of the stimulus within its receptive field, we obtain a 2D function called its receptive field profile. These turn out, for about 97% of the neurons, to be very closely described as 2D Gabor wavelets (or phasors). Some examples of empirically measured profiles are shown in the top row of the lower figure; the ideal theoretical form of such a wavelet (which we will define later) is shown in the middle row; and the difference between these two functions in the bottom row; the differences are nearly nil and statistically insignificant. So it appears that the visual cortex of the brain evolved a knowledge of the valuable properties of such wavelets for purposes of image coding and analysis!



Above: quadrature phase (90 deg) relationship between adjacent pairs of identically-tuned cortical simple cells, in response to drifting sinusoidal gratings, suggesting complex phasor processing.

Below: detailed receptive field structure of such neurons (top row); theoretical 2D Gabor phasor components (middle row); and residual differences between the data and models (bottom row).



3 Mathematical operations for extracting structure from images.

Most image processing and feature encoding operations can be interpreted, at least indirectly, in terms of 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.) should be understood at least partially in "spectral" (i.e. Fourier) terms.

In addition to this explanatory role, Fourier analysis can sometimes be used to construct useful visual representations that are invariant under translation (change in position), rotation, and dilation (change in size).

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 greyscale 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 equation 3 above: $e^{i\pi} + 1 = 0$, which also contains the five most important mathematical constants, and symbolises 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 μ_k and ν_k define the coordinates of the 2D Fourier domain. These (μ_k, ν_k) coordinates are called vector spatial frequencies, and the array of them must span the (μ, ν) Fourier plane in a uniform Cartesian lattice.

It is often useful to think of the (μ, ν) 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:

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

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 (μ_k, ν_k) vector frequency components whose associated coefficients a_k must be computed is also n x n.

3.1 Some useful theorems of 2D Fourier Analysis (background material)

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 $F(\mu, \nu)$.

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

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:

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

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.

$\boxed{\textbf{Similarity Theorem}}$**:** If the size of the original pattern $f(x, y)$ changes (shrinks/expands), say by a factor α in the x-direction, and by a factor β 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: $\frac{1}{|\alpha\beta|}F(\frac{\mu}{\alpha}, \frac{\nu}{\beta})$.

$\boxed{\textbf{Rotation Theorem}}$**:** If the original pattern $f(x, y)$ rotates through some angle θ, 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 (μ, ν) is now mapped into log-polar coordinates (r, θ) 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 θ-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).

$\boxed{\textbf{Convolution Theorem}}$**:** 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 neuron'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, where the earlier continuous integrals have now been replaced by discrete summations:

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

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 = mend; i < iend; i++) {
    for (j = nend; 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 implement the convolution in the Fourier domain because the kernel array was large, then of the four nested **for loops** in the C code above, the inner two loops would be entirely eliminated. Instead, the only operation inside the outer two **for loops** would be a multiplication:

```
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 nested two loops, 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 generally faster (implementations that make use of graphics chip hardware or employ special instruction set architectures such as SSE may involve different trade-offs).

$\boxed{\textbf{Differentiation Theorem}}$**:** 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:

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

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 ∇^2) is equivalent simply to multiplying the 2DFT of the image by a paraboloid:

$$\nabla^2 f(x,y) \equiv \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) f(x,y) \overset{2DFT}{\Longrightarrow} -(\mu^2+\nu^2)F(\mu,\nu) \qquad (12)$$

Practical Application: Multi-Resolution Edge Detection.

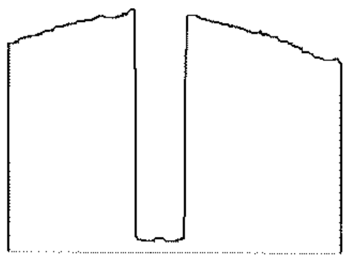
4 Edge detection operators; the Laplacian and its zero-crossings.

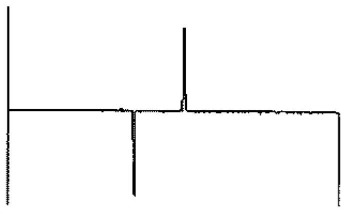
Computer vision applications often begin with feature detection, and one of the most important features to detect are edges. There are several reasons why edges are important, and why detecting the edges in a scene can be regarded as an elementary form of constructing a *signal-to-symbol converter:*

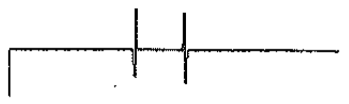
- Edges demarcate the boundaries of objects, or of material properties.
- Objects have parts, and these are typically joined with edges.
- The three-dimensional distribution of objects in a scene usually generates occlusions of some objects by other objects, and these form occlusion edges which reveal 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 organised 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, as 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] (sequence of pixels, index n), then the first-order finite difference operator (h[0],h[1])=(-1, 1) when convolved with L[n] would generate an output which is large in amplitude only where L[n] has edges (see previous figure).

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.


Original profile


Mask = [-1 1]


Mask = [1 -2 1]

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. Similarly for (-1,2,-1).

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:

-1	2	-1
-1	2	-1
-1	2	-1

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 next picture illustrates such an effect. A discrete approximation to the Laplacian operator ∇^2 in just a 3 x 3 array is:

-1	-2	-1
-2	12	-2
-1	-2	-1

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.

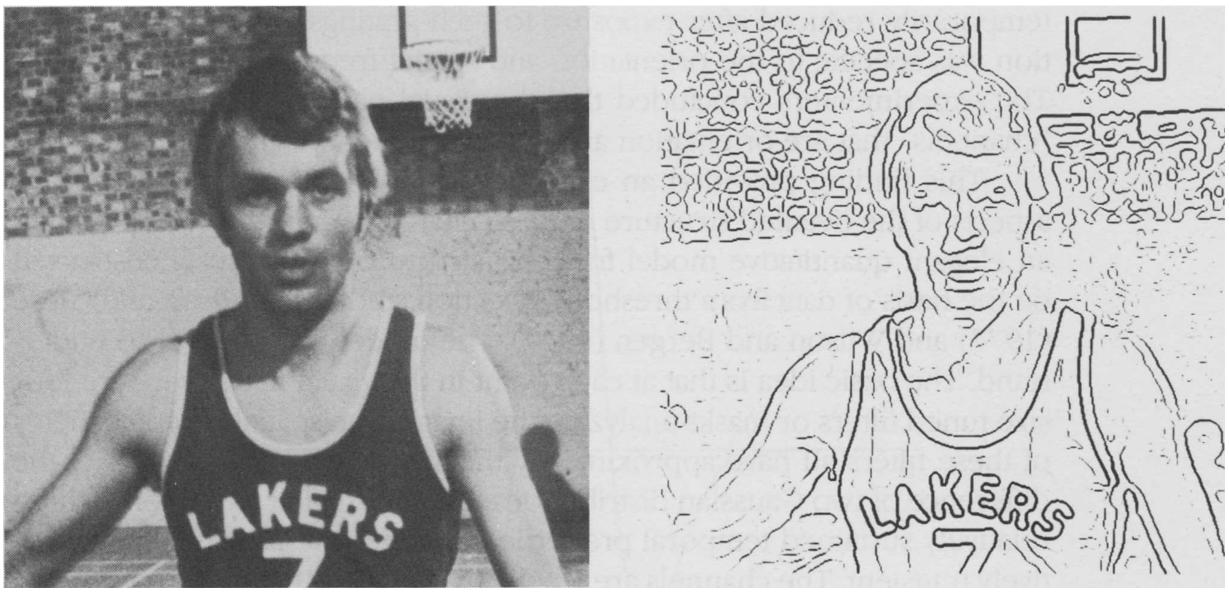


Figure 2 *Illustration of edge-detection by convolution with an isotropic Laplacian operator, and marking the zero-crossings of the result of the convolution.*

Edges in images are defined at <u>different scales:</u> 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 analysed 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 multi-scale 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. Convolving with a filter kernel that is separable is the same as convolving with two 1D kernels, one in the x-direction and another in the y-direction.
- degree of conjoint uncertainty (i.e. minimal dispersion, or variance) in the information resolved
- size of support (dimensionality of the kernel)
- preferred non-linear outputs (zero-crossings; phasor moduli; energy)
- theoretical foundations (e.g. Logan's Theorem)

4.1 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) 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 chosen scale, and hence this method is frequently exploited for edge detection in machine vision systems.

One 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 (AM) signals. For example, a pure sine wave 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 $\nabla^2 \equiv \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$ is to multiply the image spectrum by a paraboloid: $(\mu^2 + \nu^2)$. Clearly this emphasises the high frequencies at the expense of the low frequencies, and eliminates the DC component entirely (hence the output is centred around a mean of zero).

Blurring the Laplacian by a Gaussian $G_\sigma(x,y)$ of scale σ, 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 σ 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 σ 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 single 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.

As an aside, we can find an expression for $\nabla^2 G_\sigma(x, y)$ as follows. Formally, the 2D Gaussian $G_\sigma(x, y)$ is the product of two 1D Gaussians, so

$$G_\sigma(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-y^2/2\sigma^2} = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

and by doing some calculus we can show that

$$\nabla^2 G_\sigma(x, y) = \frac{\partial^2}{\partial x^2} G_\sigma(x, y) + \frac{\partial^2}{\partial y^2} G_\sigma(x, y) = \frac{x^2 + y^2 - 2\sigma^2}{2\pi\sigma^6} e^{-(x^2+y^2)/2\sigma^2} \quad (14)$$

Various representations of $\nabla^2 G_\sigma(x, y)$ (which don't need to be memorised!) are used in image processing applications.

Some open theoretical issues in this approach are:

1. It is not clear how to generalise the constraint of one-octave band-limiting to the case of 2D signals (images). E.g. should their 2DFT be confined to a "ring" 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 optimise 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 <u>could not</u> perform if our visual systems functioned in this way.

5 Multi-scale feature detection and matching.

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 σ, 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 applicability for extracting features that are more complicated (and more useful) than mere edges.

5.1 Scale-Invariant Feature Transform (SIFT).

The relative configuration of keypoint features extracted by wavelets or other multi-scale feature detectors can be used for object identification and pose extraction, with invariance to scale, orientation, affine distortion, and with some robustness to illumination.

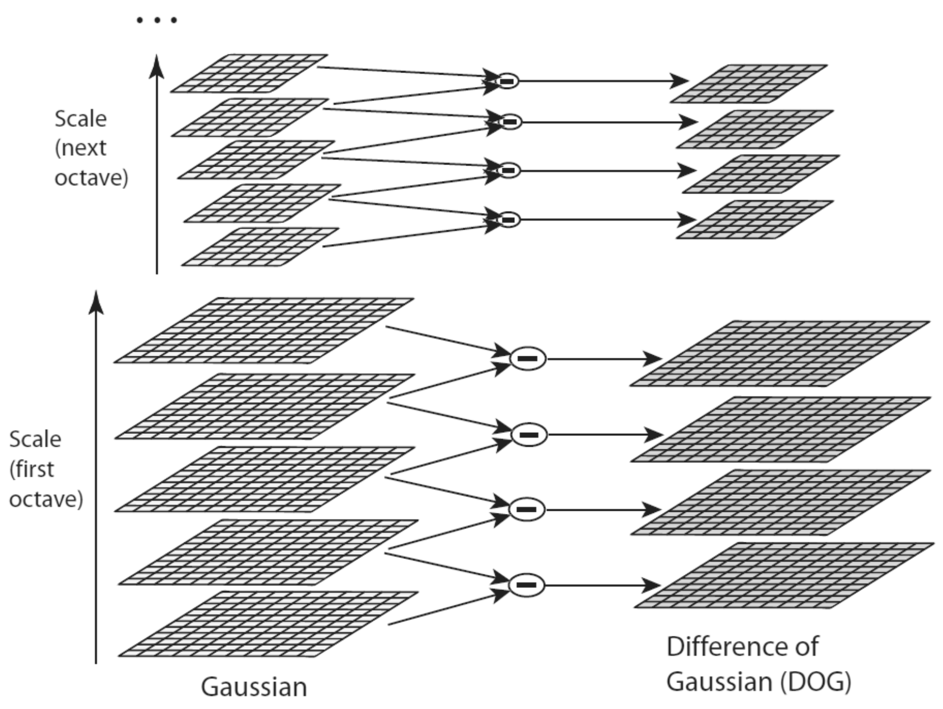


Figure 3 *Multi-scale Gaussian image pyramid representation and difference-of-Gaussian operation used by SIFT. (Lowe 2004)*

An important algorithm for detecting, describing and matching keypoint features is the Scale-Invariant Feature Transform (SIFT) developed by David Lowe. Configurations of SIFT keypoints from different images can be indexed, ordered, and statistically compared via a distance metric to find correspondences between instances of objects in different poses. The method somewhat resembles the identification of fingerprints by the relative configurations of groups of minutiae (oriented ridge terminations, bifurcations, spurs, etc) but is done across different scales of a Gaussian pyramid. Pose invariance is achieved by clusters of features "voting" on the most plausible pose, i.e. the object and pose consistent with the most features, even allowing for projective affine distortions from rotation in depth.

To find stable features that are invariant to scale, SIFT uses a Gaussian scale-space approach. Keypoints are detected by first finding scale-space extrema. This is achieved by convolving the image with Gaussian filters G at different scales of analysis σ and differencing the resulting blurred images at neighbouring scales (σ and $k\sigma$ for some constant k) to find local minima and maxima. Formally, this *difference-of-Gaussian* operation DoG on image data I is given by

$$DoG(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

where we consider the 2D Gaussian G to be a function of three variables rather

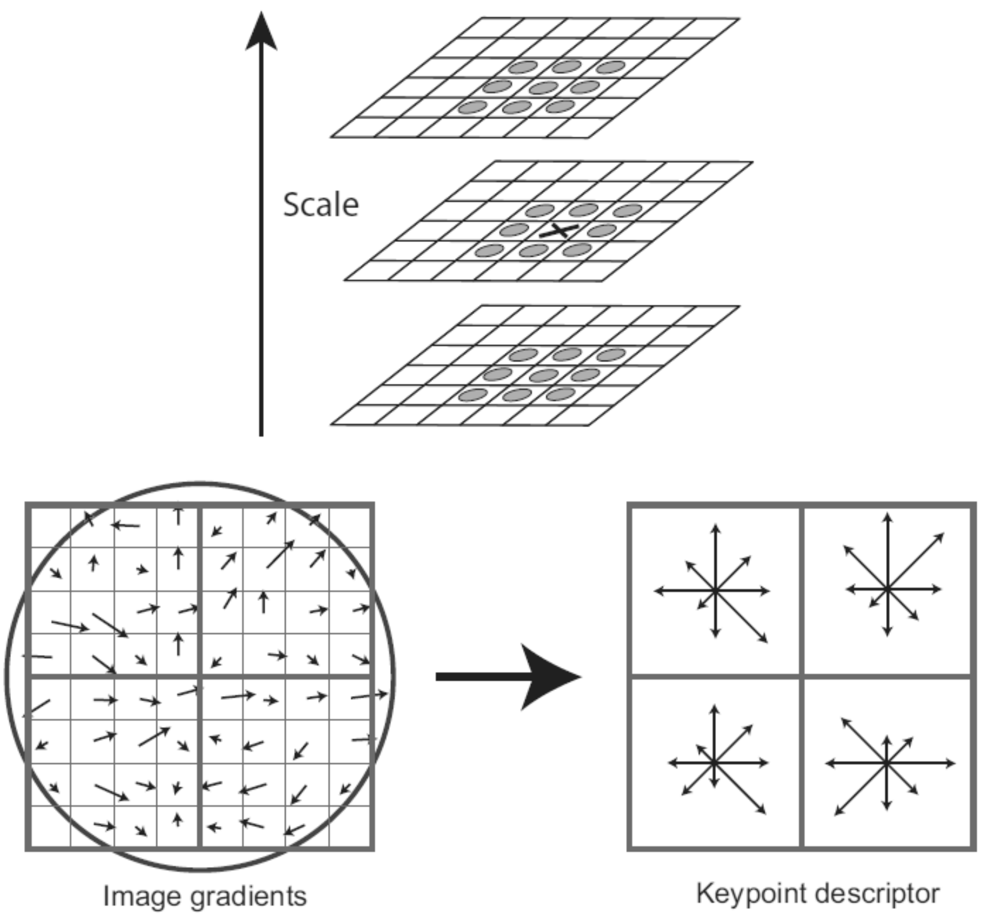


Figure 4 *Top: Pixel locations considered by SIFT when selecting extrema of difference-of-Gaussian images. Bottom: SIFT keypoint descriptor based on local gradient orientation histograms. (Lowe 2004)*

than just two:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

The motivation for this is that DoG is a good approximation to a scale-normalised Laplacian of Gaussian $\sigma^2\nabla^2 G$, which has been shown to provide scale invariance (Lindberg 1994) and whose extrema yield stable image features. It can be shown that

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$$

If we consider the finite difference approximation to $\frac{\partial G}{\partial \sigma}$ at neighbouring scales $k\sigma$ and σ

$$\frac{\partial G}{\partial \sigma} \approx \frac{G(x,y,k\sigma) - G(x,y,\sigma)}{k\sigma - \sigma}$$

then by multiplying by $k\sigma - \sigma = (k-1)\sigma$ we get

$$G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2 \nabla^2 G \tag{15}$$

Therefore $\sigma^2\nabla^2 G$ can be implemented efficiently by applying the difference-of-Gaussian operation to a "Gaussian image pyramid" where neighbouring levels correspond to versions of the image blurred by Gaussians whose scale factors differ by a constant k. Note that we now have *two* parameters that determine the scales of analysis involved in feature detection:

- The σ of the Gaussian filters smoothes the image by blurring it, which helps to eliminate noise but also eliminates detail (low-pass filter in the Fourier domain). Convolution with a Gaussian followed by re-sampling is the standard technique for downsampling images, for reasons discussed at the start of this section.
- The constant k is a multiplicative factor between neighbouring Gaussian-blurred images whose difference we wish to compute to extract stable features. SIFT does this by comparing each pixel in the *DoG* images to its eight neighbours at the same scale and nine corresponding neighbouring pixels in each of the adjacent scales (pyramid levels). A pixel is selected as a candidate keypoint if its value is the maximum or minimum among all compared pixels, as shown in figure 4.

The actual choice of values for σ and k is driven by theoretical analysis (i.e. the quality of the approximation in equation 15), empirical analysis, and efficiency considerations. SIFT starts with an initial σ_0 and considers several octaves i of scale-space such that $\sigma_{i+1} = 2\sigma_i$. After each octave i, the image is down-sampled by a factor of 2 in each dimension for efficiency. However, if we took difference-of-Gaussian images only after every octave, then k would be equal to 2, which clearly makes the approximation in equation 15 rather inaccurate. SIFT therefore subdivides each octave into s intervals, and since we want σ to double after that many intervals, it follows that

$$k = 2^{1/s}$$

and the value of σ at octave i and interval n of the pyramid is given by

$$\sigma(i,n) = \sigma_0\, 2^{i+n/s};\ n \in [0, s-1]$$

A value of $s = 3$ was found by Lowe to provide a good accuracy vs efficiency trade-off. The number of octaves depends on original image resolution. Figure 3 illustrates this approach.
We will not discuss the other steps of SIFT in detail. SIFT performs interpolation to localise candidate keypoints with sub-pixel accuracy and discards keypoints with low contrast or stability. In order to achieve invariance to rotation, a keypoint descriptor based on local gradient directions and magnitude is used (see figure 4). The descriptor is invariant to image rotations since the bins of the orientation histograms are normalised relative to the dominant gradient orientation in the vicinity of the keypoint.

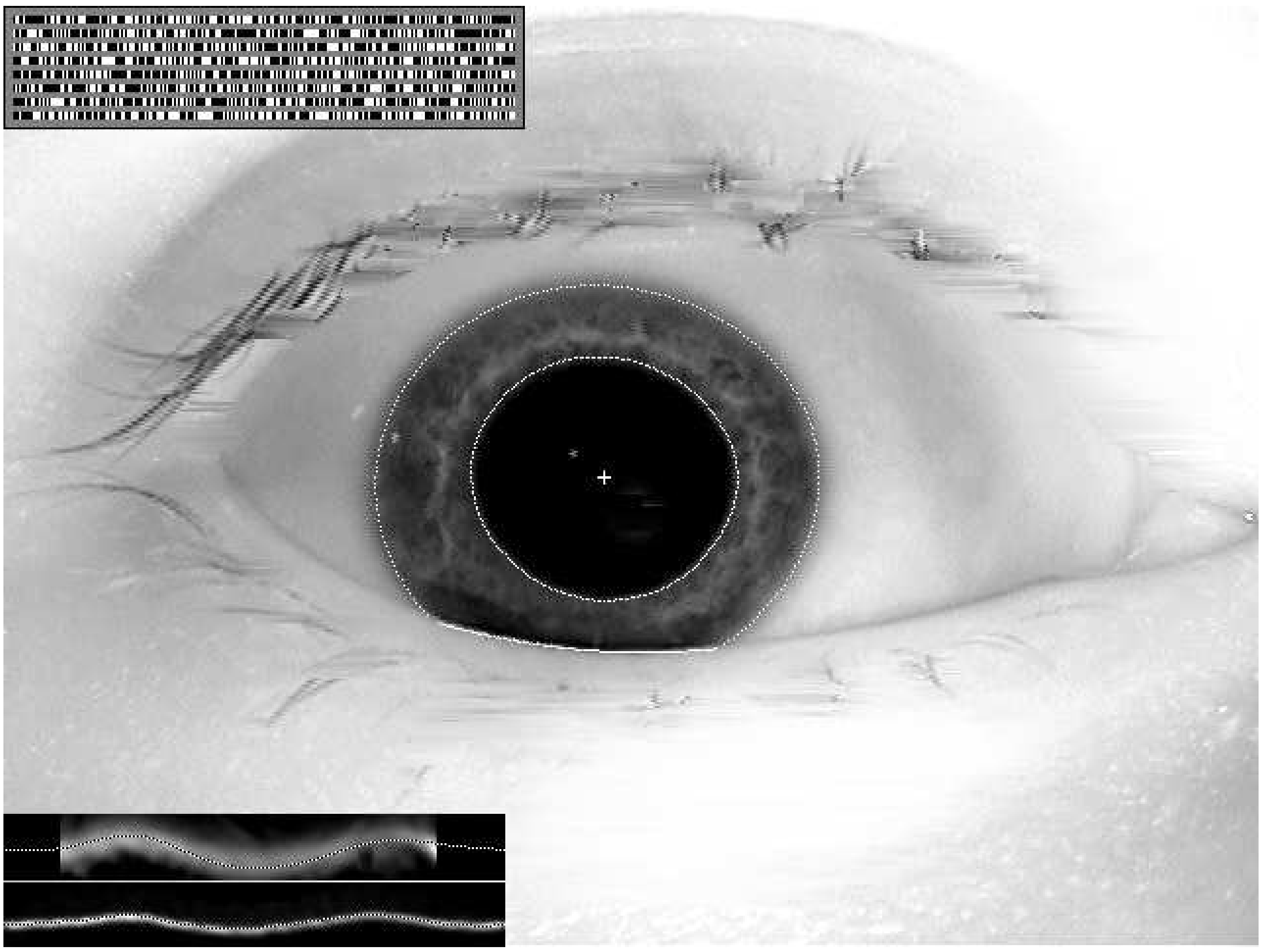


Figure 5 *Active contours are deformable yet constrained shape models. The "snakes" in the box show radial edge gradients at the iris boundaries, and active contour approximations (dotted curves).*

5.2 Active Contours ("snakes"). Fourier boundary descriptors.

The detection of edges and object boundaries within images can be combined with constraints that control some parameters of admissibility, such as the shape of the contour or its "curvature energy," or the scale of analysis that is being adopted. These ideas have greatly enriched the old subject of edge detection, whilst also enabling the low-level operators we have considered so far to be directly integrated with high-level desiderata about shape, such as geometry, complexity, classification and smoothness, and also with theory of evidence and data fusion. The image of the eye shown in figure 5 (illustrating *Iris Recognition,* a technology for biometric automatic identification of persons) contains three active contours: two defining the inner and outer boundaries of the iris, and one defining the boundary between the iris and the lower eyelid. All three are determined by the same general methods. Evidence for local edge structure is integrated with certain constraints on the boundary's mathematical form, to get a "best fit" that minimises some energy function or other "cost" function.

Thus we have the combination of two factors: a *data term* and a *cost term* (the latter sometimes also called a *smoothness term* or an *energy term*), which are in contention, in the following sense: we could fit the available edge data with arbitrarily high precision, if we used a model with enough complexity; but simpler models are generally more useful and credible than overly complex models (which "over-fit" the data). For example, the basic outline of a person's hand consists of a simple form having 5 semi-parallel appendages for fingers. How much more detail is needed, in order to detect and classify such generic shapes as hands? Greater detail might fail to be satisfied by many valid cases. So the cost term acts to keep the model simple, e.g. by penalising excessive kinks in it when seeking consistency with the data.

When shape description or pattern recognition is formulated in terms of the above two factors, the solution is often obtained by regularisation methods. These are iterative numerical methods for finding a set of model parameters that minimise (or optimise) a *functional* that is a linear combination of the two terms, with some trade-off parameter λ for specifying their relative importance. Effectively these methods convert our problem into one of calculus:

$$\arg\min \int \left((M-I)^2 + \lambda (M_{xx})^2\right) dx$$

where M is the shape model, and I is the image data (reduced here to a single dimension x for simplicity). The first term inside the integral seeks to minimise the squared-deviations between the model and the image data. If this were the only term, then a closed-form solution could be found when the model is just some linear combination of functions such as polynomial or Fourier components, requiring only matrix (linear algebraic) operations to estimate the "least-squares" parameters of the model. But the constraints imposed by the second ("smoothness") term cause the model to be more or less flexible, i.e. more or less willing to bend itself to fit every detail of the data, by penalising the sum of squared second derivatives. Parameter λ gives us a knob to turn for setting how rigid or flexible our active contour snake should be.

The behaviour of these operators for contour detection and description were illustrated by the white outline graphics in the eye image shown in figure 5. The eyelid boundary is generated by a low-order polynomial spline. The iris inner and outer boundaries are generated by Fourier series expansions constrained to fit the data "snakes" shown in the lower left corner, which would be perfectly straight and flat if these boundaries of the iris could be described simply as circles. The *IrisCode* is iris demodulation using 2D Gabor wavelets.

5.3 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 localised at position x_0, complex modulated at frequency μ_0, and with size or spread constant α. 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 $f(x)$ centred on the origin ($x_0 = 0$), its Fourier Transform $F(\mu)$ is simply a Gaussian centred 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 μ_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 parameterised 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 analysing 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.

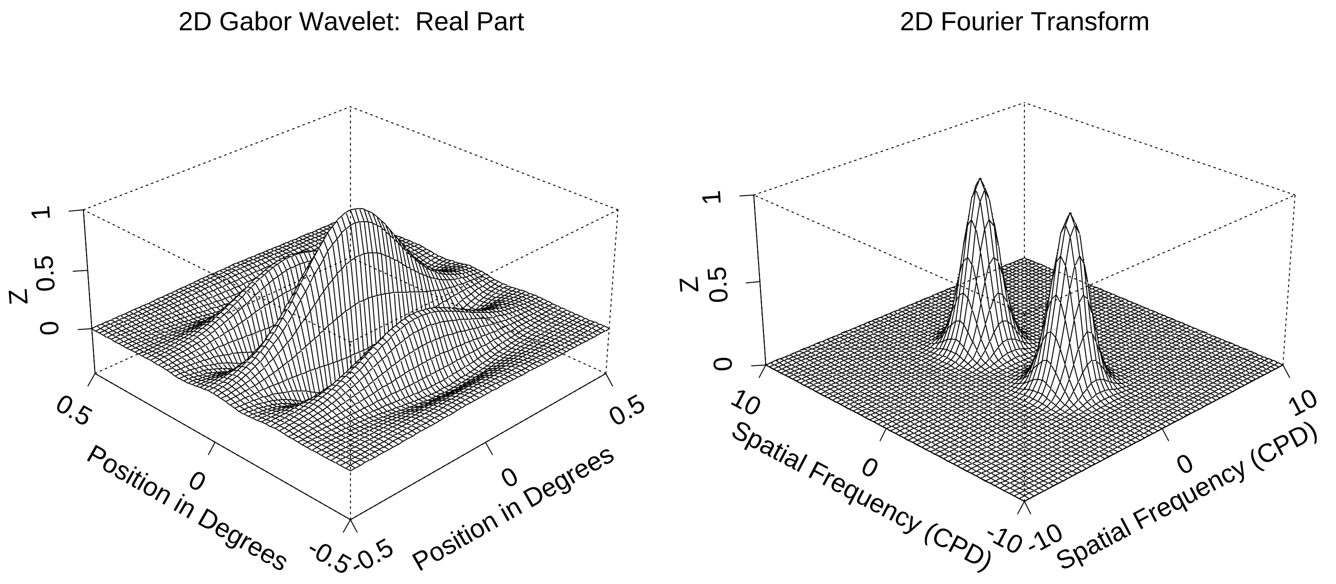


Figure 6 *The real part of a 2D Gabor wavelet, and its 2D Fourier transform.*

5.4 Generalisation of wavelet Logons to 2D for image analysis

An effective method for extracting, representing, and analysing 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 neurons 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").

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, (α, β) 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, centred at the origin $(x_0, y_0) = (0, 0)$ and with unity aspect ratio $\beta/\alpha = 1$ is shown in figure 6, together with its 2D Fourier transform $F(u, v)$.

By appropriately parameterising 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 θ:

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

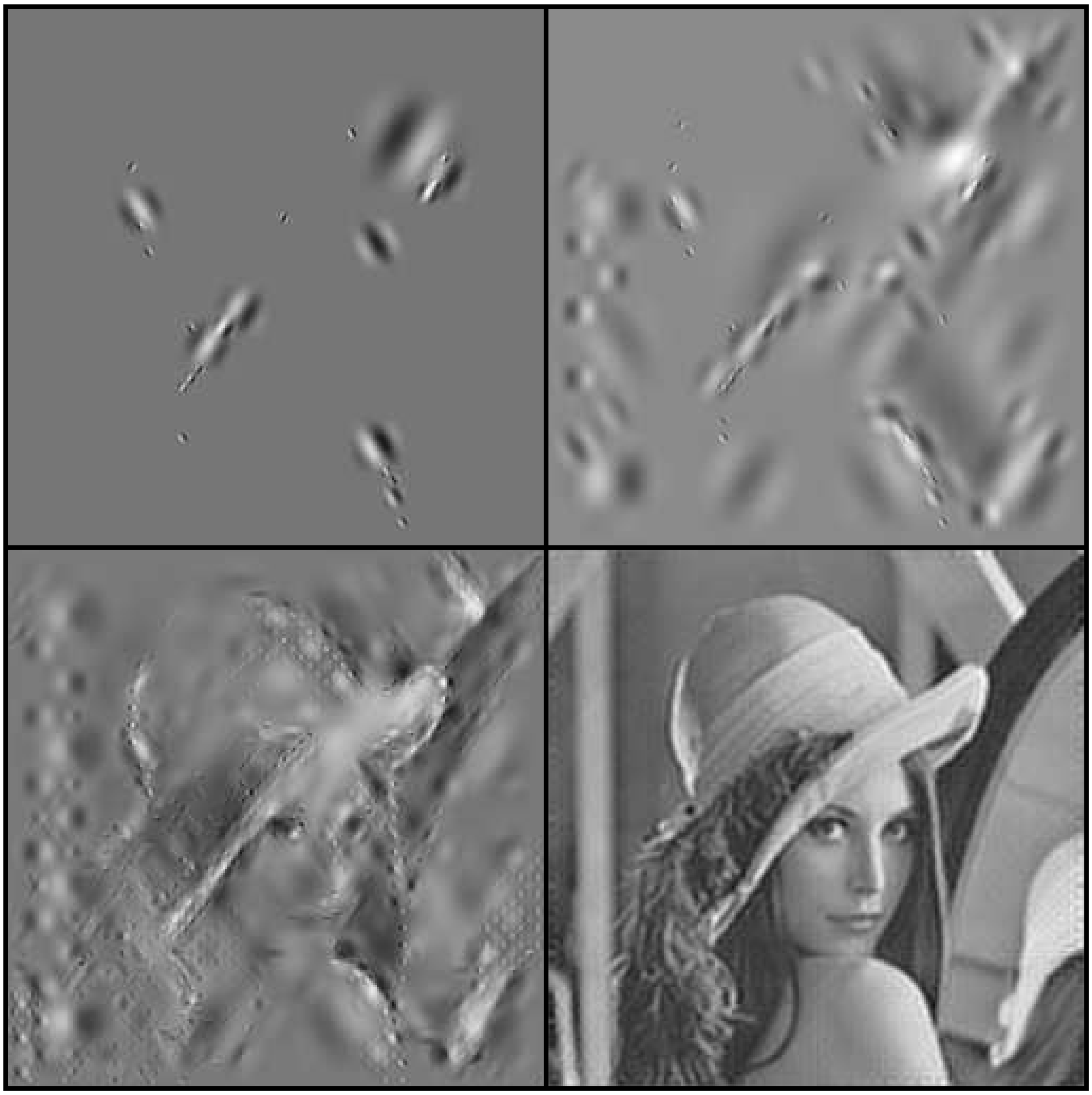
$$y' = 2^{-m}[-x\sin(\theta) + y\cos(\theta)] - q$$

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 can be illustrated by reconstruction of a facial image, in stages. (See the example in figure 7 of image reconstruction in stages.) Note how efficiently the facial features, such as the eyes and mouth, are represented using only a small number of wavelets.

5.5 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



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

Figure 7

"Gabor domain" of representation actually embraces and unifies both of these other two domains. How?

In the wavelet equations above, the scale constant α (and β 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 recognising image structure, merely by changing a single scaling parameter.

Aristotle defined vision as “knowing what is where.” We have noted the optimality (conjoint uncertainty minimisation) 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 neurons 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 (quadrature-paired means that the wavelets are offset by 90 degrees in phase).

6 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.

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.
3. Properties of the camera (or viewer), such as (i) geometry and viewing angle; (ii) spectral sensitivity; (iii) prior knowledge, assumptions, and expectations.

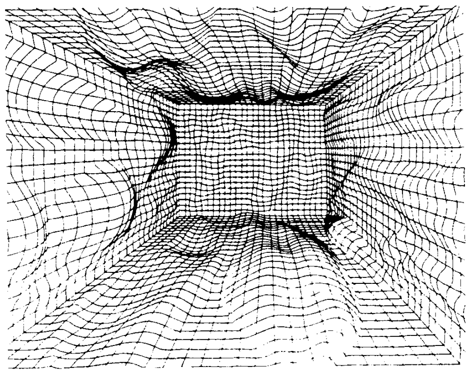
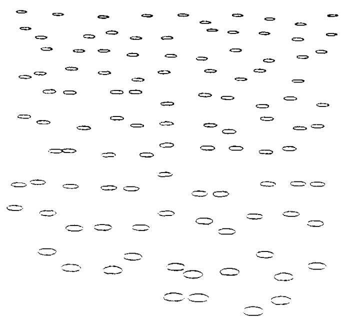
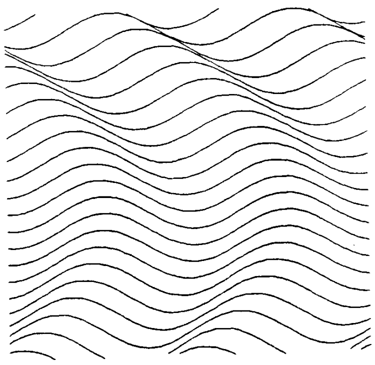
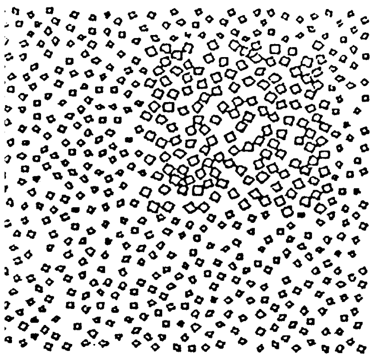
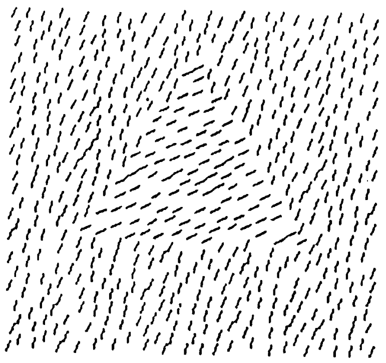
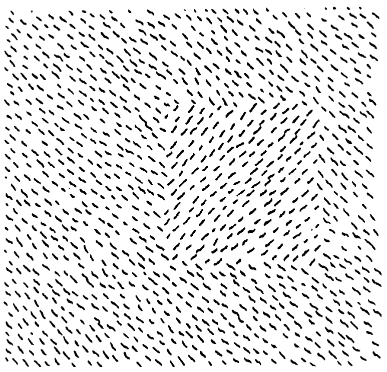
The aim of this lecture is to study how these many factors can be disambiguated and even exploited, in order to try to make objective inferences about object and world properties from these ambiguous and confounded image properties.

6.1 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 <u>image segmentation</u> 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?

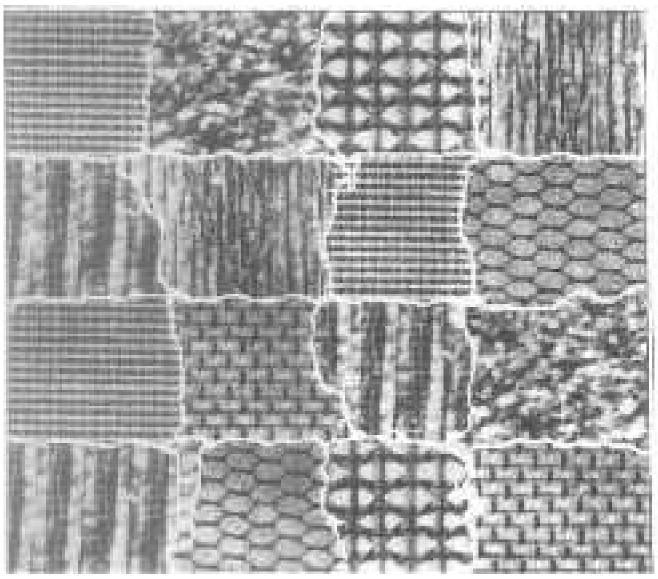







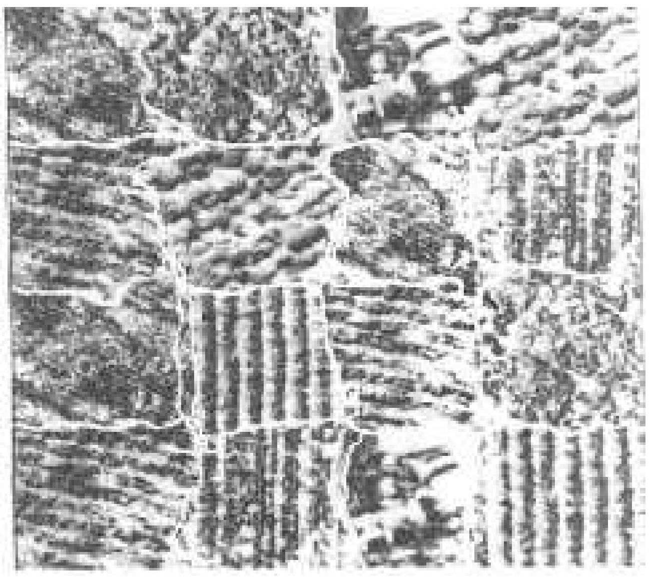
As implied by the root of the word, which links it to textiles, texture is defined by the existence of certain statistical *correlations* across the image. These can be almost anything, from quasi-periodic undulations as one might see in water ripples or in woven fabrics, to repetitive but spot-like features. Many natural scenes, such as woodlands, grasslands, mountain ranges and other terrains, have such properties which give them a distinctive identifying visual signature. Many natural textures can appear to be almost fractal, i.e. self-similar across different scales. 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 <u>local</u> regional information as a basis for texture-based image segmentation.

Hence the ideal solution is to "window" the sinusoids so that they analyse 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 localisation is greater than that of any other function. As discussed in section 5, the multiplication of a complex exponential with a bivariate Gaussian produces a *2D Gabor wavelet*. The pictures below illustrate successful segmentation of collages of textured natural scenes, as well as of textured artificial objects, using 2D Gabor wavelets for local spectral analysis to infer and measure their textural discriminators.





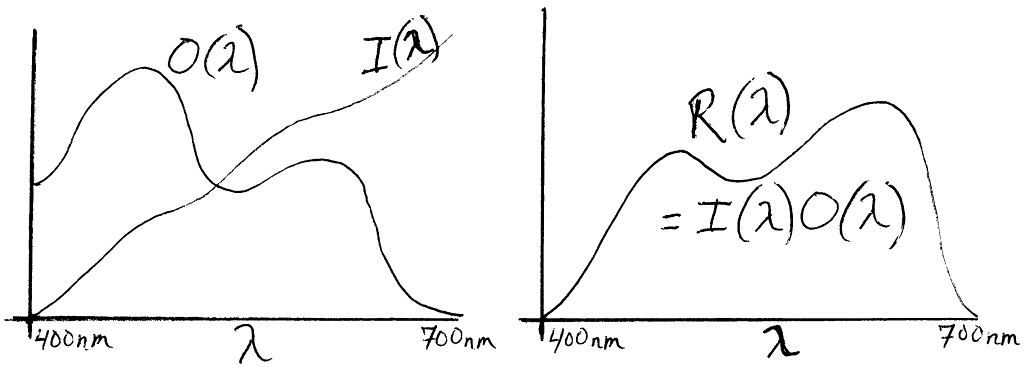
6.2 Colour information.

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 λ, 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 λ.
- 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.

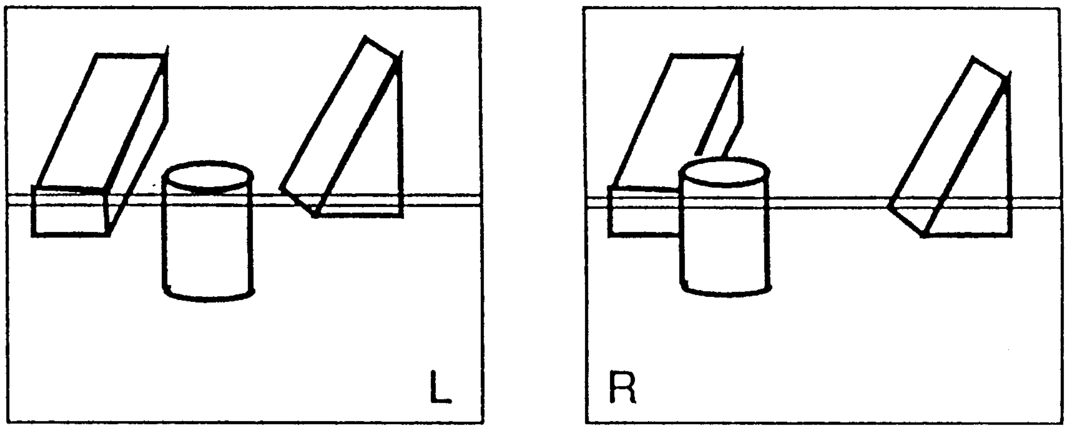
The second observation is that even when the wavelength composition of light reflected from each object is exactly the same (i.e. the three light sources are adjusted separately for each object to ensure that the light reflected in the three wavebands as measured by a spectral photometer is exactly the 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 inferred from the available retinal measurements $R(\lambda)$ without explicitly knowing $I(\lambda)$.

6.3 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.*



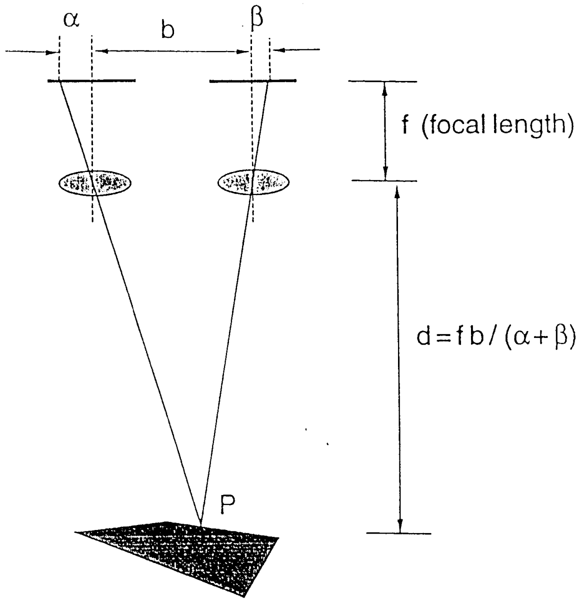
This "error signal" becomes greater in proportion to the distance of the object in front or behind the point of fixation, and so it can be calibrated to obtain a depth cue. It also becomes greater with increased spacing between the two eyes or cameras, since that is the "base of triangulation." (That is why WWI armies 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, unconstrained 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 maximise its efficiency. In initially sparsely sampled (coarsely blurred and under-sampled) images, the permutation-matching space of possible corresponding points is greatly attenuated 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 above simplifying diagram, 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 α in one image and β in the other image, then the distance d to the object in front of the two lenses is simply:

$$\boxed{d = fb/(\alpha+\beta)}$$

6.4 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" can be measured in the two image frames once their backgrounds have been aligned. This 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 <u>velocity vector field</u> 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 and temporal integration windows). This is known as the *aperture problem*.
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.

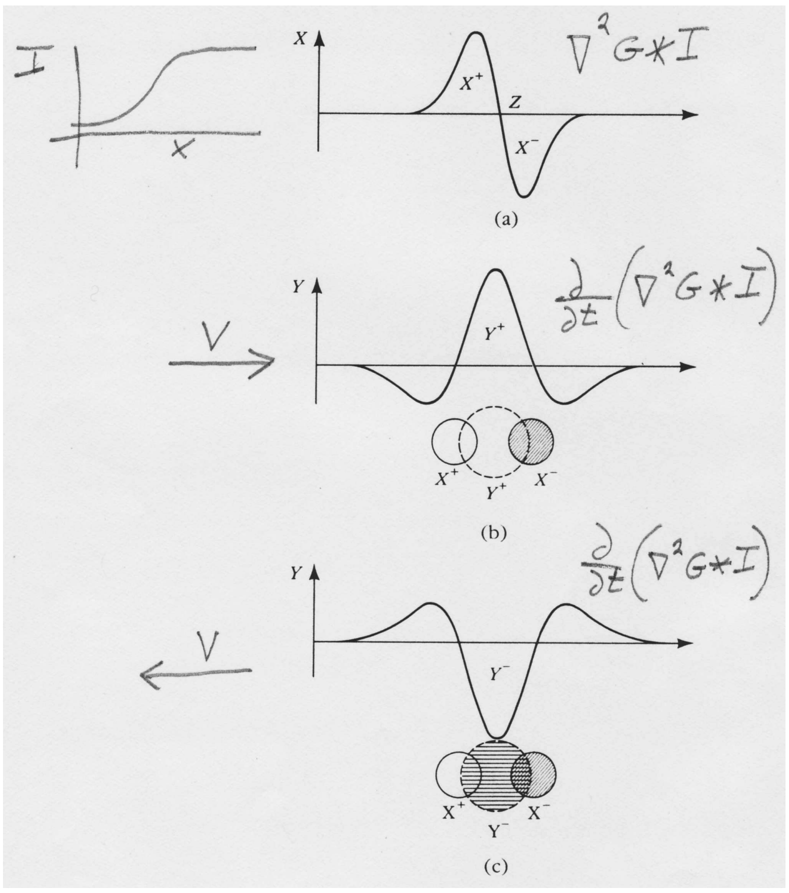
The major classes of models and approaches to motion detection are largely inspired by detailed neurobiological studies of motion processing both in the invertebrate eye and in mammalian retina and cortex. Diverse mathematical frameworks have been proposed, but the main classes of models are:

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 $\vec{v}$:

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

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).



I
X
X
X^+
Z
X^-
$\nabla^2 G * I$
(a)
Y
V
Y^+
$\frac{\partial}{\partial t}(\nabla^2 G * I)$
X^+
Y^+
X^-
(b)
Y
V
Y^-
$\frac{\partial}{\partial t}(\nabla^2 G * I)$
X^+
Y^-
X^-
(c)

Dynamic Zero-Crossing Models .

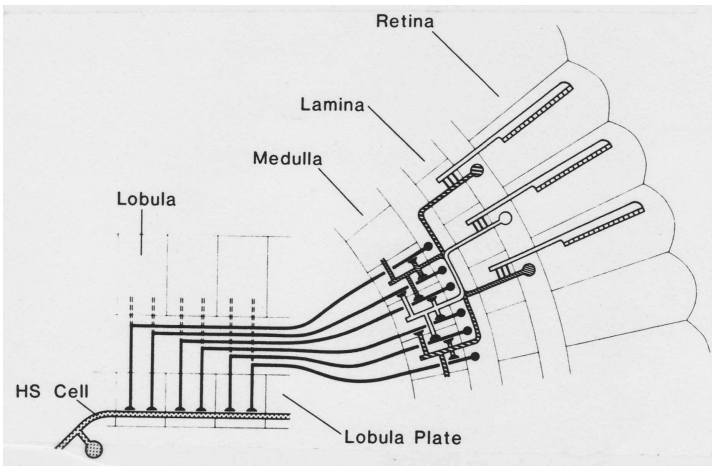
Measure image velocity by first finding the edges and contours of objects (using the zero-crossings of a blurred Laplacian operator!), and then take the time-derivative of the Laplacian-Gaussian-convolved image:

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

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 τ. Finding the pair of these intervals which maximises 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 which we desire to know.



Detailed studies of fly neural mechanisms (above) for motion detection and visual tracking led to elaborated correlation-based motion models.

Spatio-Temporal Spectral Models .

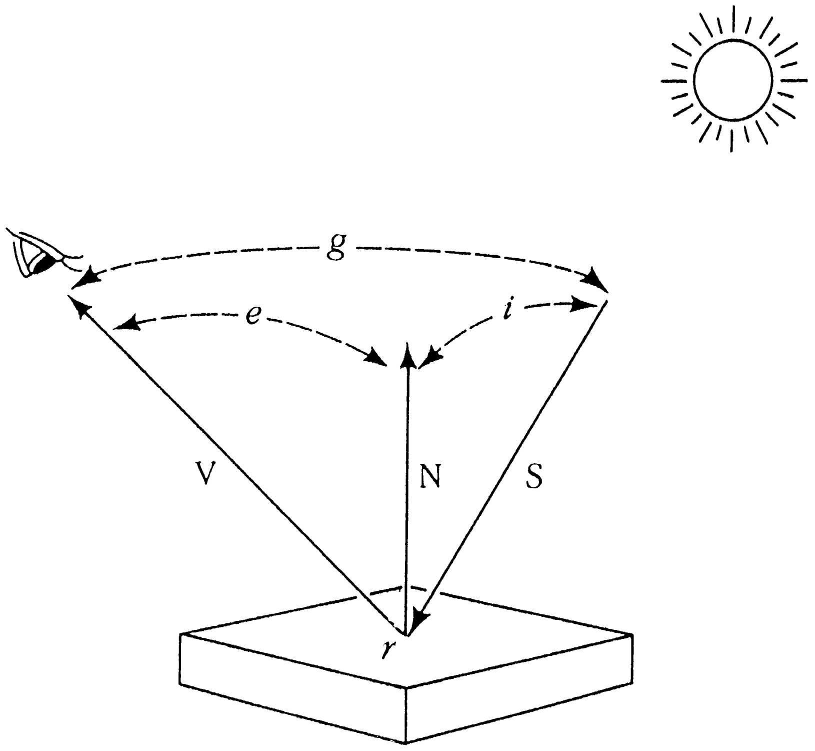
It is possible to detect and measure image motion purely by *Fourier* means. This approach exploits the fact that motion creates a covariance in the spatial and temporal *spectra* of the time-varying image $I(x, y, t)$, whose three-dimensional (spatio-temporal) Fourier transform is defined:

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

In other words, rigid image motion has a 3D spectral consequence: the local 3D spatio-temporal spectrum, rather than filling up 3-space $(\omega_x, \omega_y, \omega_t)$, collapses onto a 2D 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 centre frequencies are **co-planar** in this 3-space are activated together. This is a consequence of the **Spectral Co-Planarity Theorem**, which states that translational image motion of velocity $\vec{\mathbf{v}}$ has a 3D spatio-temporal Fourier spectrum that is non-zero only on an inclined plane through the origin of frequency-space. Spherical coordinates of the unit normal to this spectral plane correspond to the speed and direction of motion.

7 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:



The definitions of the angles i, e, and g

- 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, projection screens,...

  A Lambertian surface looks equally bright from all directions: the amount of light reflected depends only on the angle of incidence i of the illuminant, not on the angle of emission e.

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.

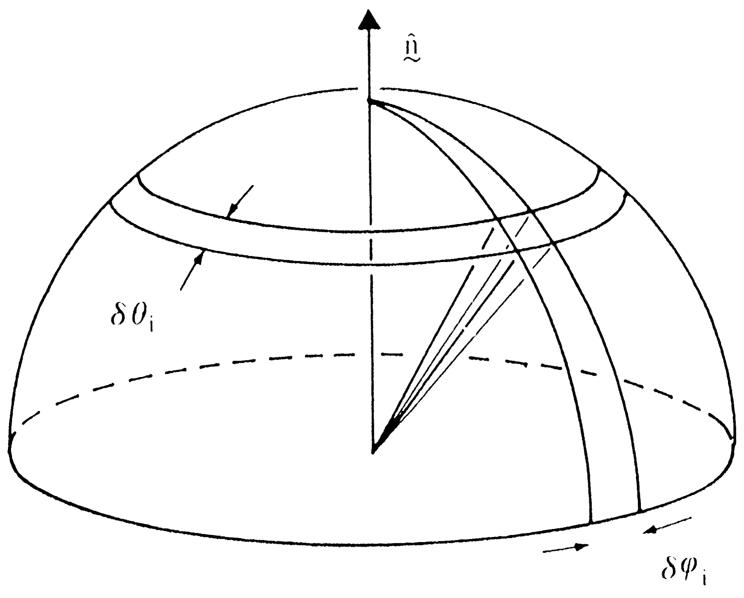
- A specular surface is locally mirror-like. It obeys Snell's law (i.e. the angle of incidence of light is equal to the angle of reflection from the surface), and does not scatter light. Most metallic surfaces are specular.
- The reflectance map is a function $\phi(i, e, g)$ which relates intensities 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 (i.e. flat) 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 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



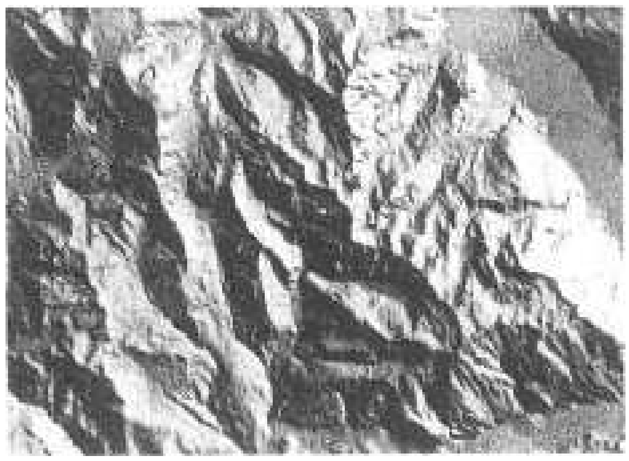
reflect and scatter light, is to build in certain assumptions about the nature of the surfaces from other kinds of evidence. This requires us to consider the general problem of inference and integration of evidence.

8 Shape description. Codons, superquadrics and surface geometry.

Just as illustrated earlier by the examples of inferring 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")

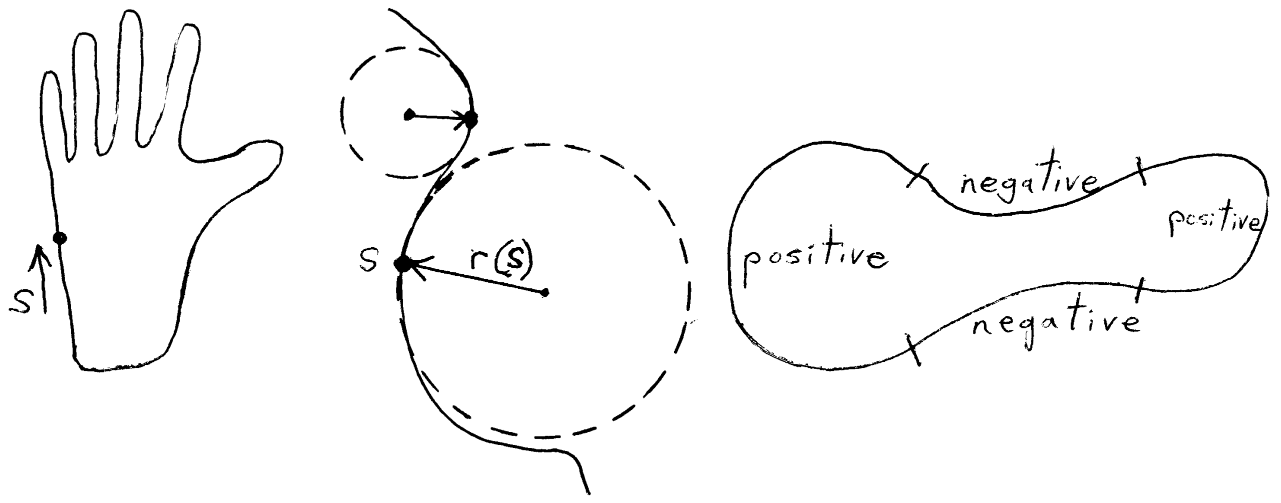


The inference of a surface shape (a relief map, or an object-centred description of a surface) from shading information is an inherently ill-posed problem because the data necessary for the computation is simply not known. One has to introduce ancillary assumptions about the surface material composition, its albedo and specularity parameters, the illumination of the scene and its geometry, before such inferences become possible. It is almost as though the assumptions are more important than the available image data. The computational nature of the inference task then becomes one of *constraint satisfaction*, and solving such a problem is often formulated as an optimisation or *relaxation problem.* Often there are rival alternative solutions. In human vision these can be triggered and alternated (e.g. converting a crater into an apparent mound, reversing the inferred surface shape) simply by changing a cue about the direction of illumination.

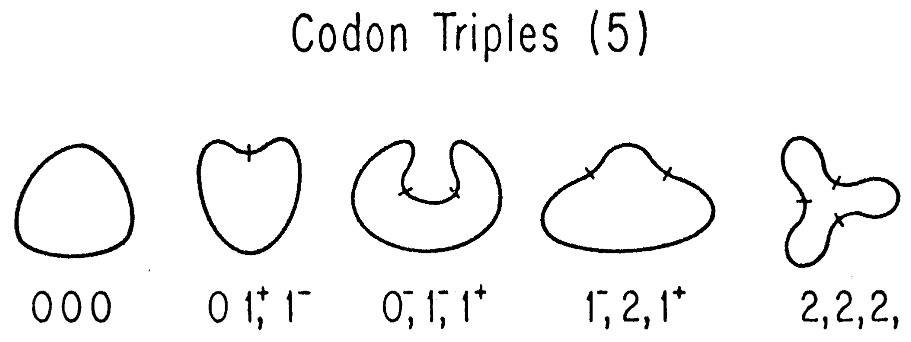
8.1 How should shape be represented? Boundary descriptors; codons.

Closed boundary contours can be represented completely by their curvature map $\theta(s)$ (the reciprocal of the local radius of curvature $r(s)$ as a function of position s along the contour). This is the *Fundamental Theorem of Curves.* Local radius of curvature $r(s)$ is defined as the limiting radius of the circle that best "fits" the contour at position s, in the limit as the arc length Δs shrinks to 0, and the local curvature of the contour at that point is:

$$\theta(s) = \lim_{\Delta s \to 0} \frac{1}{r(s)}$$



Closed boundary contours can be expanded with basis functions (such as "Fourier descriptors" of the radius of curvature) from their curvature map, in order to generate a shape description that is invariant to translation, rotation, and dilation. By cataloguing 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 recognised. Interestingly, Logan's Theorem (about the richness of zero-crossings for capturing bandlimited 1D signals completely) arises again in this context: the curvature map of a closed contour is a bandlimited signal, and it can be described by its zero-crossings; such a description amounts to a shape classification. This is one of several approaches proposing an elementary grammar for shape, and it can be generalised to surfaces.



The curvature map $\theta(s)$ together with a "starting point" tangent $t(s_o)$ specifies a shape fully. Some nice properties of curvature-map descriptions are:

1. The description is position-independent (i.e., object-centred).
2. The description is orientation-independent (rotating the shape in the plane does not affect its curvature map).
3. The description represents mirror-symmetric shapes simply by a change in sign:
$$\theta(s) \rightarrow \theta(-s)$$
4. Scaling property: Changing the size of a shape simply scales $\theta(s)$ by the same factor. The zero-crossings are unaffected.
$$\theta(s) \rightarrow K\theta(s)$$

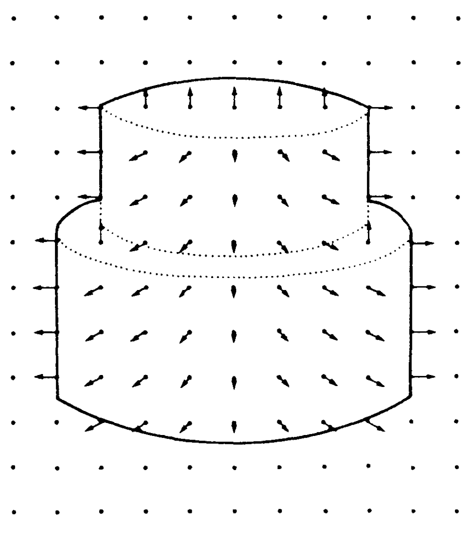


Figure 8

8.2 The "2.5-dimensional" sketch

A scheme which David Marr proposed for bridging the gap between 2D image (appearance-based) descriptions and 3D model-based descriptions is called the "2.5-dimensional sketch." Surface normals are computed and assigned to each point in the image domain, which indicate 3D shape information. Looking at such "pin-cushion" diagrams (figure 8) does effectively convey three-dimensional shape.

8.3 3D Object-centred coordinates. Superquadrics.

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

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

Examples include "generalised cylinders" and cubes (large exponents); prolate spheroids (rugby balls) 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-centred, "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.

8.4 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-7 standard for motion image encoding specifies provision for both facial animation and body animation.

9 Perceptual psychology and visual cognition. Visual illusions.

Opportunities to learn from biological visual systems, for the design of artificial ones, are not limited to low-level neural mechanisms. Insights from *perceptual and cognitive psychology* are also relevant.

9.1 Perceptual organisation.

In earlier sections we investigated various mechanisms for image analysis, such as filters for spatial forms and edge detectors, and special mechanisms for handling texture, colour, stereo, and motion information. But how does all of this get "put back together again" into a unified visual percept?

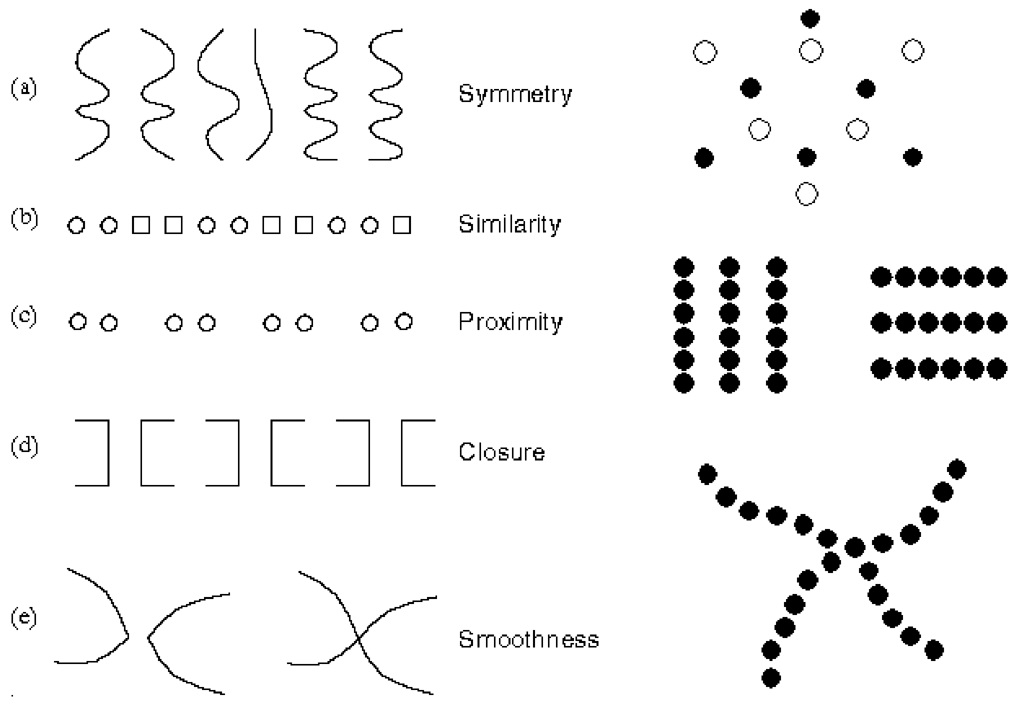


Figure 9 *Some Gestalt laws of perceptual organisation.*

This was a motivating question for a school of research in visual perception called *Gestalt Psychology,* associated mainly with Koffka, Köhler, and Wertheimer in Germany in the 1930s. "The whole is greater than the sum of its parts" might almost be a slogan for the Gestaltists, who sought to understand how meaningful wholes *(Gestalten)* are constructed in a way that seems (introspectively) to precede the analysis of parts and properties. Thus was born the study of perceptual organisation. The Gestaltists enumerated a series of principles collectively called the *Law of Prägnanz* (the law of conciseness or salience) that seemed to underlie our perceptual organisation of form and patterns, based on sub-laws of grouping by proximity, similarity, "good continuation", and closure (the filling-in of missing parts). While they offer useful principles (see figure 9 for some examples) by which visual percepts can be grouped together, Gestalt theories of perceptual organisation are descriptive rather than explanatory, and therefore don't provide detailed help in designing artificial vision systems. However, as we will discuss in the next two lectures, many object recognition systems implicitly need to perform perceptual grouping to recognise "wholes from their parts" by identifying and combining relevant features at different scales of analysis and by incorporating domain specific prior knowledge. This process is made explicit in parts-based recognition systems for the detection of people, animals, and vehicles. As Koffka noted in 1935:

> *"...to apply the Gestalt category means to find out which parts of nature belong as parts to functional wholes, to discover their position in these wholes, their degree of relative independence, and the articulation of larger wholes into sub-wholes."*

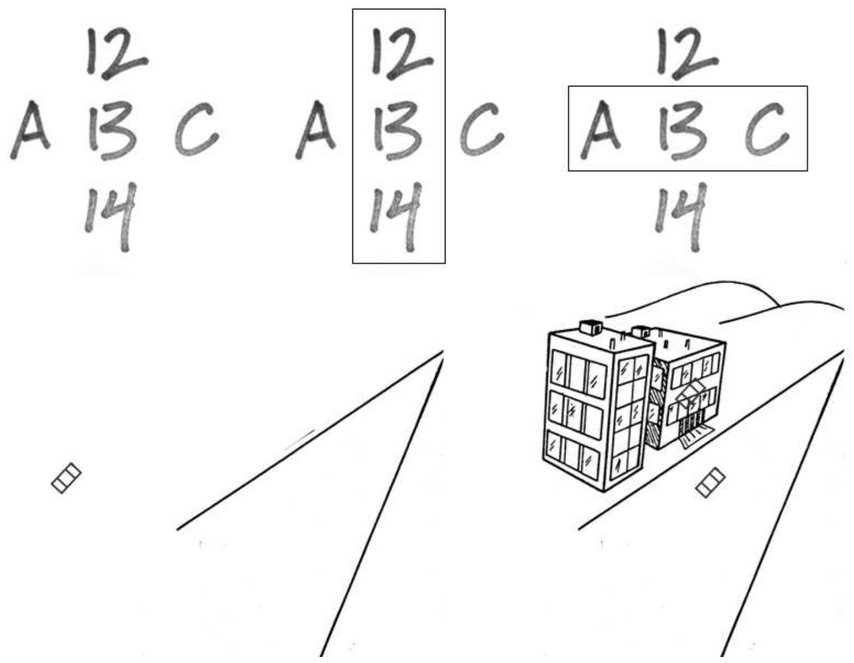


Figure 10 *The importance of context. Top: the same shape is interpreted differently depending on its surroundings. Bottom: individually ambiguous shapes have a clear interpretation when seen together.*

9.2 Vision as language processing

We discussed previously that the challenge of vision can be described in terms of building a *signal-to-symbol converter*. This gives rise to the view that (high-

level) vision may be regarded as closely related to language processing. Finding symbolic interpretations of underlying signal data needs to incorporate a notion of the "grammar" (syntax) and "meaning" (semantics) governing a particular visual task so that the most likely explanation of the observed data can be found. Processing may then be performed selectively in response to queries formulated in terms of the structure of the domain, i.e. relating high-level symbolic representations to extracted visual and temporal features in the signal.
More recent developments include the idea of a *process grammar* which models objects and shapes in terms of their morphogenesis (the likely sequence of steps in their evolution from simpler forms). A common theme is that vision is inference, going well beyond the given.

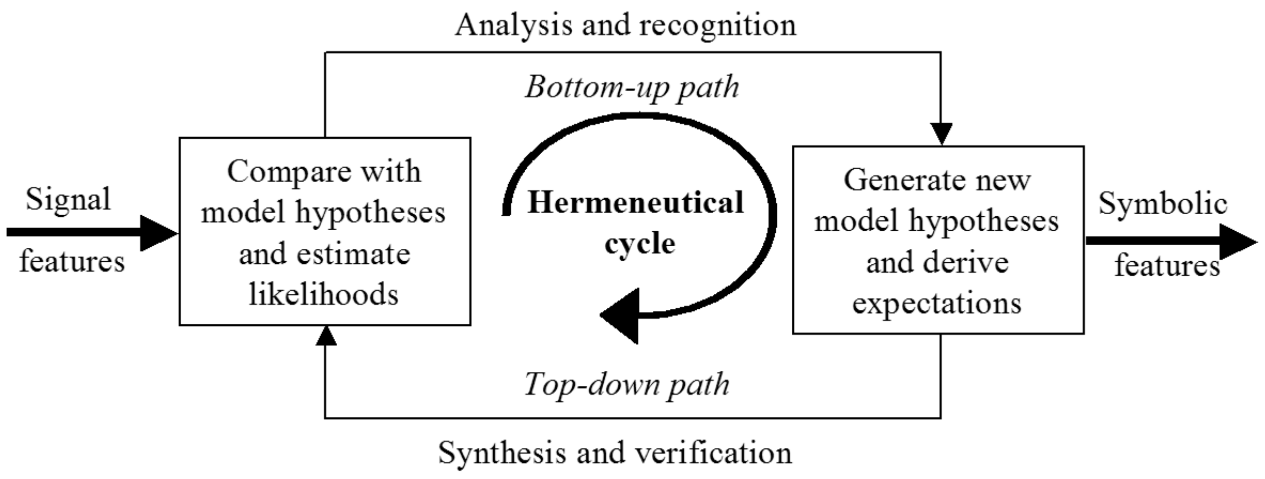
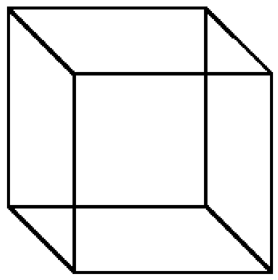

Analysis and recognition
Bottom-up path
Signal
features
Compare with model hypotheses and estimate likelihoods
Hermeneutical cycle
Generate new model hypotheses and derive expectations
Symbolic
features
Top-down path
Synthesis and verification

Figure 11 *The Hermeneutical cycle for iterative interpretation in a generative (hypothesise and test) framework.*

9.3 Vision as perceptual inference

Objects are not always unambiguously describable solely on the basis of their constituent parts, as the Gestaltists also recognised. Object recognition is often dependent on *context* (see figure 10). In the next lecture we will discuss how vision can be approached as knowledge-driven perceptual inference, and how Bayesian and statistical techniques form an important basis for modelling contextual knowledge and performing vision as inference from the data.
Such ideas have a rich heritage in Artificial Intelligence, although early AI approaches to perceptual inference were often very "brittle" and limited to toy problems. The advent of machine learning and robust probabilistic techniques is beginning to change this, although most vision systems are somewhat piecemeal in that they are limited to very specialised tasks such as recognising a

particular type of object in (usually static) images. A more ambitious aim would be a dynamic goal-directed vision system capable of iteratively comparing low-level visual percepts with high-level models to derive new hypotheses about the world. These can in turn guide the search for evidence to confirm or reject the hypotheses on the basis of expectations defined over lower level features. Figure 11 illustrates this approach.
An aspect of perceiving scenes as meaningful wholes, as opposed to an atomistic, literal, or elemental description in terms of individual features, is the grouping of features into a 3D model. A classic illustration of this idea of "vision as model-building" is the Necker cube: a set of 12 planar line segments that are always seen as a 3D solid (a cube); yet having two conflicting (bistable) visual interpretations:



Such bistable percepts are examples of perceptual rivalry: two or more alternative ways to interpret the same visual stimulus. Several more examples are given in figure 12: Ruben's vase/faces; girl/witch; man/rat. The key notion is that *percepts are hypotheses*: they are top-down interpretations that depend greatly on contexts, expectations, and other extraneous factors that go beyond the actual stimulus.

In the examples in figure 13, illusory contours are perceived in places where no actual contours exist. In the upper examples the illusory contours even seem to demarcate a region that is "brighter" than its surrounds; and the illusory contours can even take curvilinear trajectories. But defined by what??

In the lower pair of examples, the circle and the square appear significantly deformed in shape by their context. Are such aspects of your visual system "bugs," or "features?" Should such inaccuracies in representing patterns also be designed into machine vision systems, intentionally or epiphenomenally?

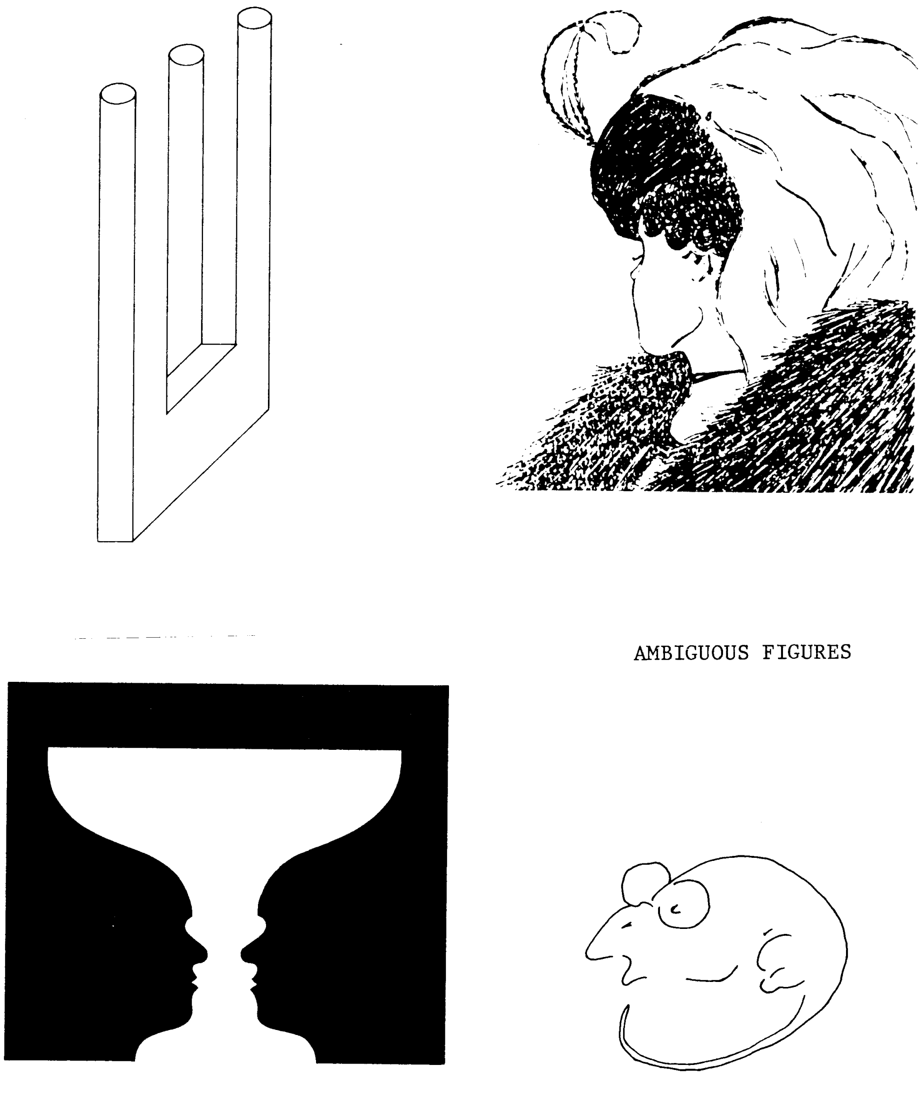


Figure 12

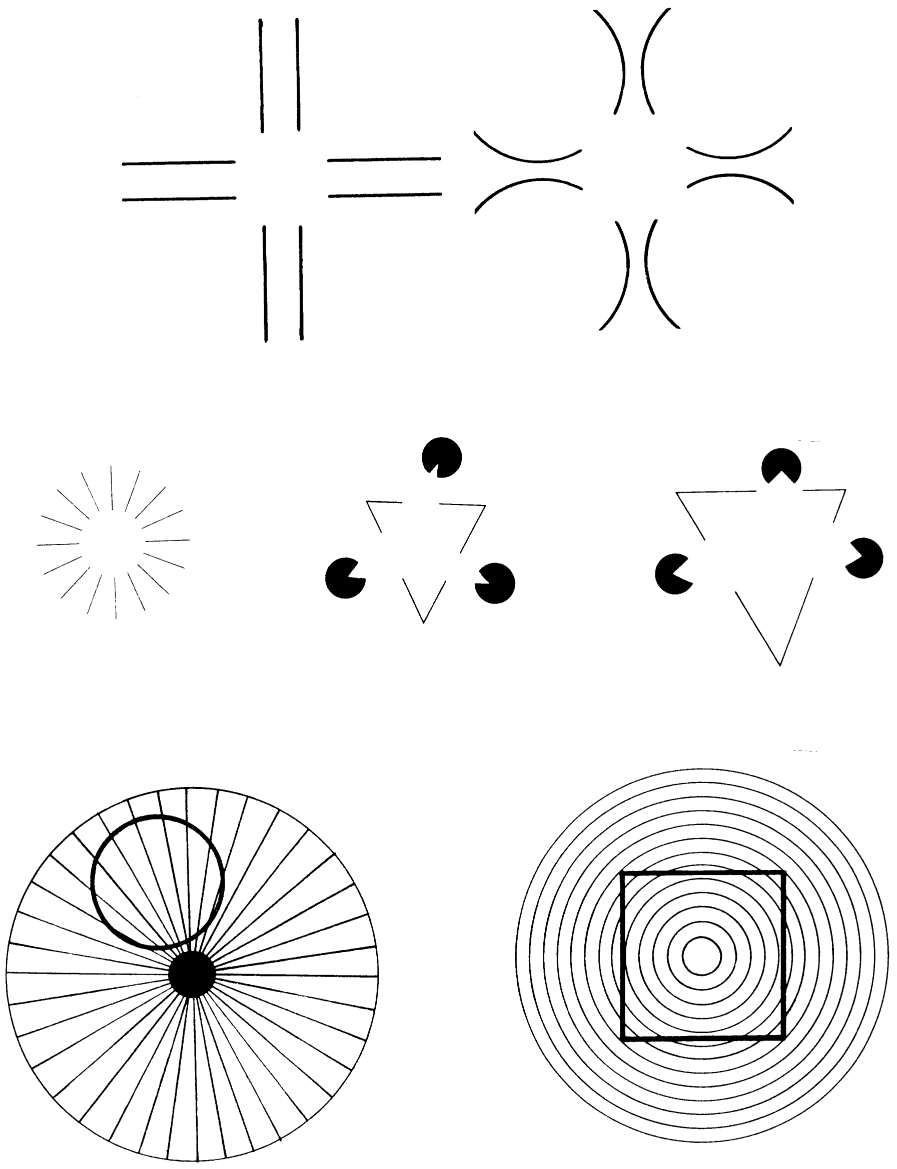


Figure 13

9.4 Visual illusions.

Let us examine specific illusions of geometry, size, brightness, and motion. As always, the question for us to ask in connection with Computer Vision is, what do these illusions reveal about visual mechanisms? Should we try to design our algorithms in such a way that they too would "suffer" from such illusions, at least as an epiphenomenon [side-effect] of the desired functionality?

The pattern on the left in figure 14 has a set of long oblique lines which are in fact parallel, but they appear very non-parallel. Why does the presence of the other short lines so severely distort our judgement of orientation and parallelism? In the visual cortex, there are both competitive and cooperative neural processes, operating over both orientation and proximity; presumably these create the illusion seen. In the examples on the right of figure 14, the

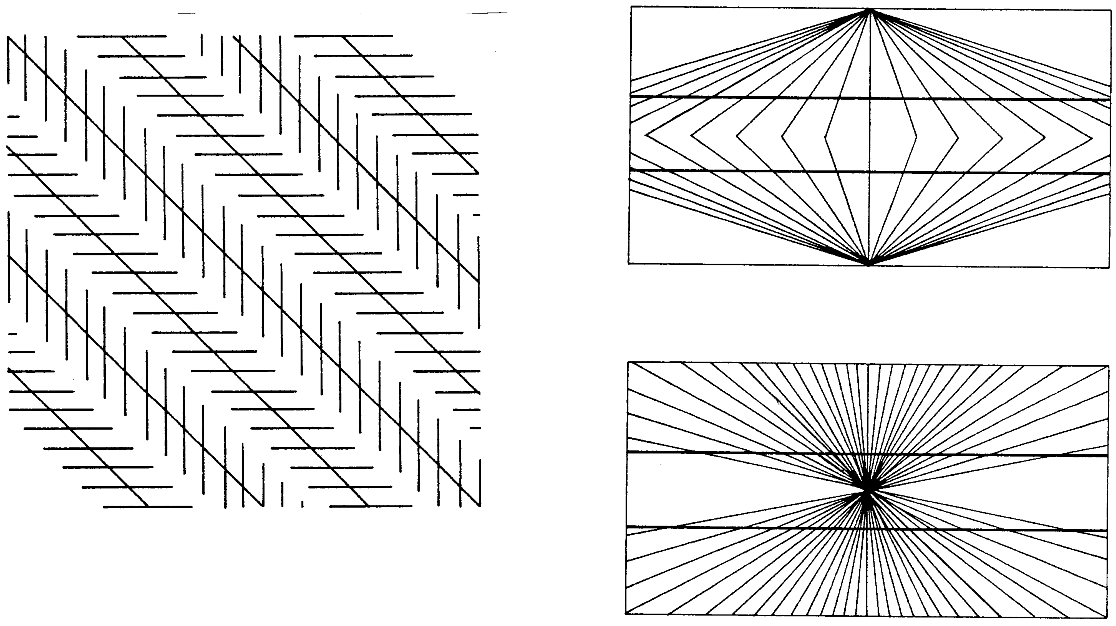


Figure 14

net effect is a continuous bending of the (in fact straight and parallel) lines, bowing them together in one case and apart in the other.

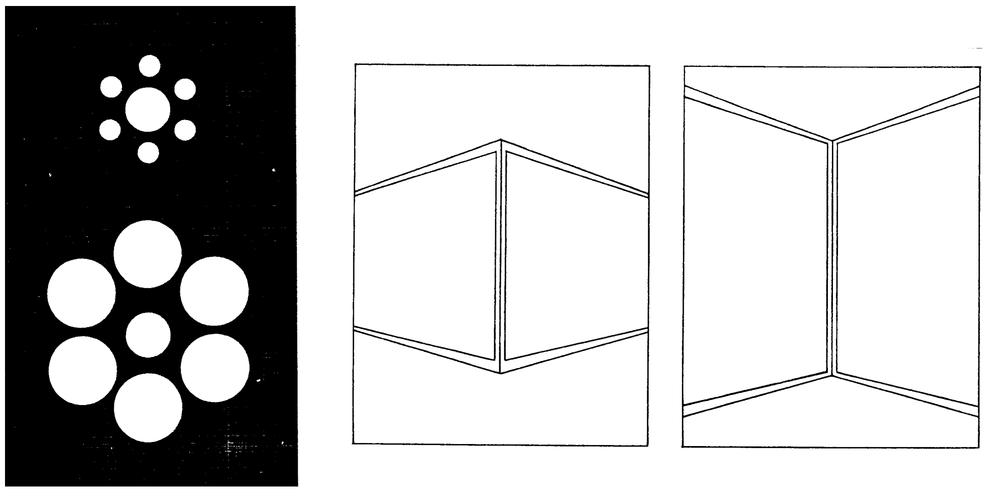


Figure 15

Competition appears also in the domain of size, as seen in the example shown in figure 15 on the left: the central disk in fact is the same size in both cases. The right is an illustration of the Müller-Lyer illusion: the vertical segment is in fact the same length in both cases, contrary to appearances. But there it is the presence of the other oblique lines that somehow cause the illusion.

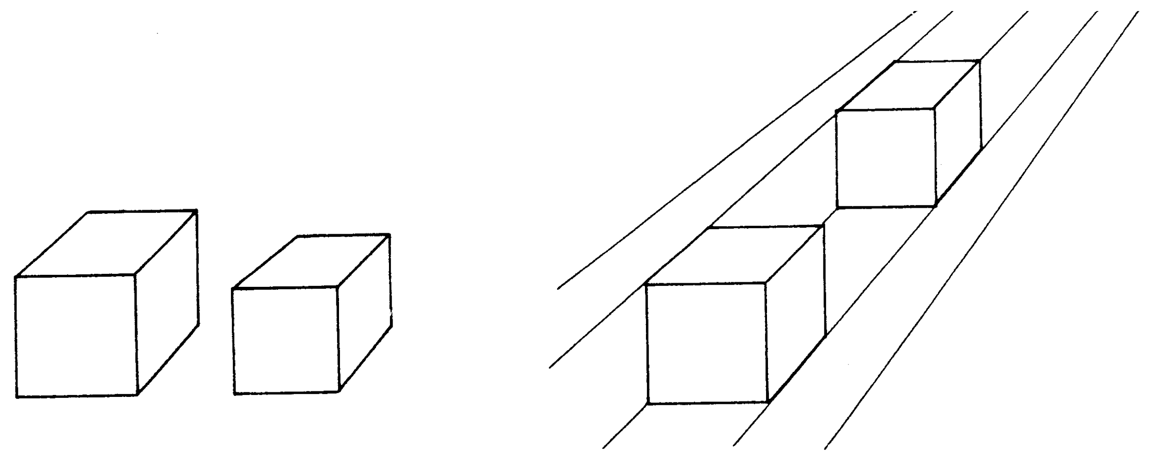
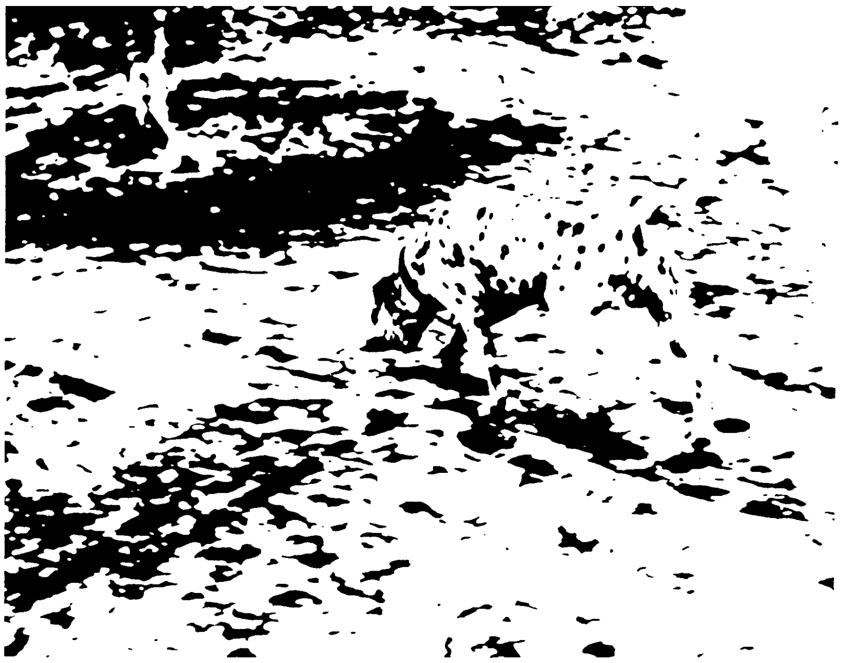


Figure 16

Finally, the presence of extraneous cues plays a large role in many inferences. The two boxes on the left in figure 16 are clearly of different sizes; yet when forced to see them as displaced in depth, we judge them to be in reality the same size. [Illusions of brightness and motion can be found on the course website.]

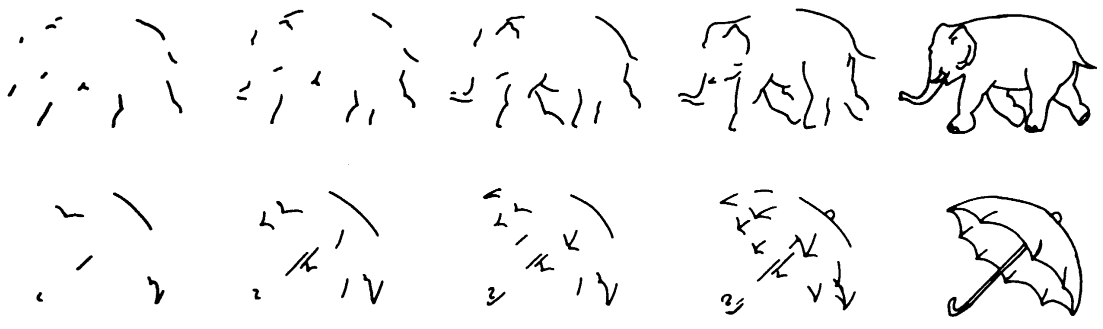
10 Bayesian inference in vision. Classifiers; probabilistic methods.

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 recognising them
- the unpredictability of the future
- the fact that objects and events have associated likelihoods of occurrence (depending on context)
- the uncertainty of causation
- 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. The framework to adopt is that, in a sense, *the world consists of probabilities,* and that visual processing really amounts to *computing probabilities and assigning them.*

Examples of decisions-under-uncertainty in vision:

- Medical diagnosis; radiology: Is this a tumour? 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 arrives gradually over time? How should a-priori knowledge about the probabilities of events in the world be combined with available incoming data?

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

10.2 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), or of achieving fusion amongst different and possibly incommensurable forms of data, is that of Bayesian inference.

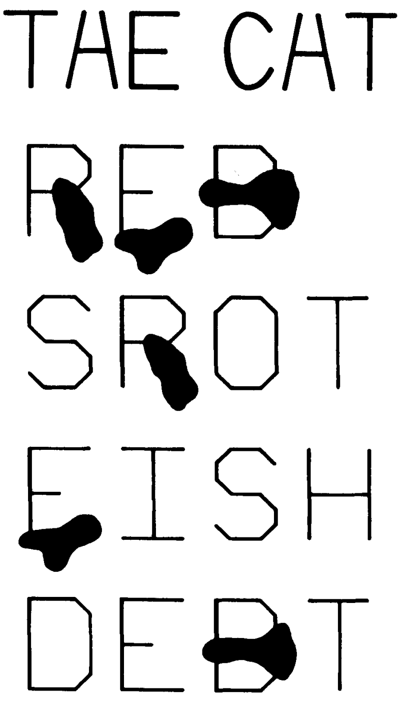
Bayesianism interprets probability as "degree-of-belief," rather than as "frequency of occurrence," and argues for weighing all evidence with all possible (imaginable) interpretations and their associated (estimated) probabilities. Bayes' rule, named after the 17th-century cleric, Thomas Bayes, is a 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. Some aspects of Bayesian interpretation in vision are evident in the way we read the following texts, in which the same letter stimulus is read in completely different ways depending on local context:

We begin with an informal statement of Bayes' rule for drawing inferences from data. If H represents a hypothesis about the "state of the world" (e.g. the object in an image) and D represents the available image data, then the explanatory conditional probabilities $p(H|D)$ and $p(D|H)$ are related to each other and to their unconditional likelihoods $p(H)$ and $p(D)$ as follows:

$$p(H|D) = \frac{p(D|H)p(H)}{p(D)} \tag{16}$$

The probabilities in this equation are often referred to using the following terminology:

$$posterior = \frac{likelihood * prior}{evidence}$$



For example, a human agricultural expert, or an artificial expert system, has knowledge of the form $p(D|H)$: Given a plant (or a hypothetical disease state) H, there is a corresponding conditional probability $p(D|H)$ of observing certain image data D. However, typically the goal of computer vision and pattern recognition is to calculate just the *inverse* of that conditional probability: given evidence in terms of image data D, what is the probability $p(H|D)$ that the hypothesis (of plant or disease state H) is true?

Bayes' rule (equation 16) specifies the formal procedure for calculating such inferences $p(H|D)$, given the observations, the unconditional probabilities, and the prior expert agricultural knowledge $p(D|H)$. It thereby offers a clean and simple interface between a knowledge base and visual data. A key feature of Bayes' Theorem is that it provides a mechanism for repeatedly updating our assessment of a visual hypothesis as more data arrives incrementally. 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 facilitates 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 decision environment with four possible outcomes:

1. Hit (True accept): Actually same; decision "same".
2. Miss (False reject): Actually same; decision "different".
3. False Alarm (False accept): Actually different; decision "same".
4. Correct Reject (True reject): Actually different; decision "different".

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

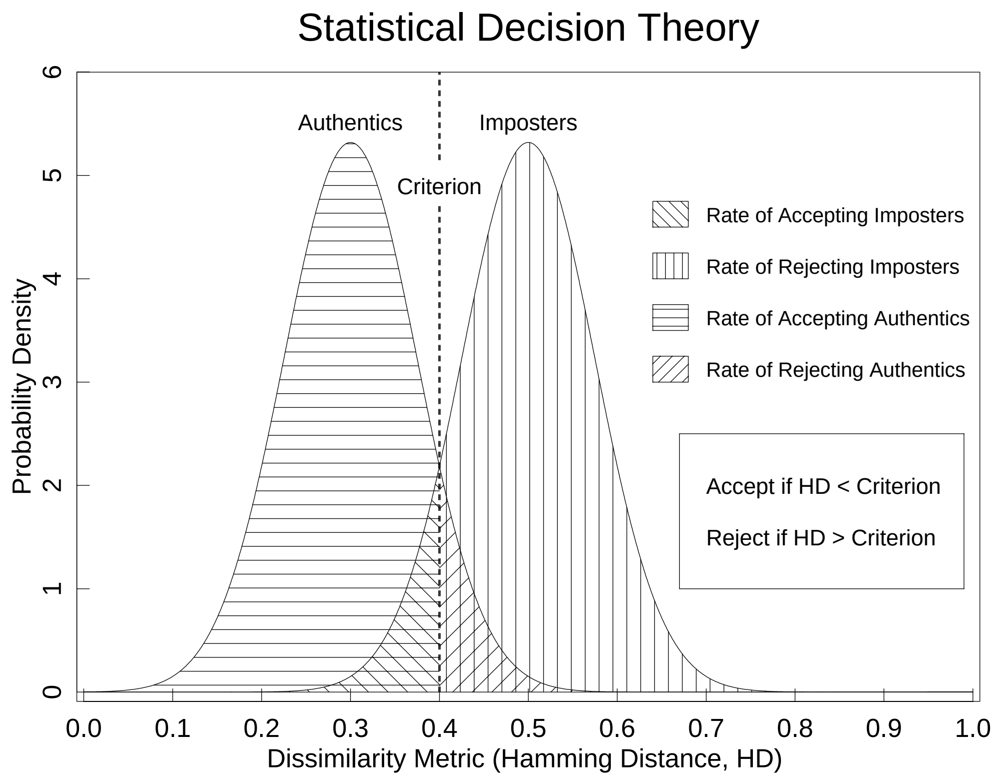


Figure 17 *Two-choice Decision Environment of Statistical Decision Theory*

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 <u>coupled</u> 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 optimise 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 <u>Signal Detection Theory</u>, or also as <u>Statistical Decision Theory.</u> 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).

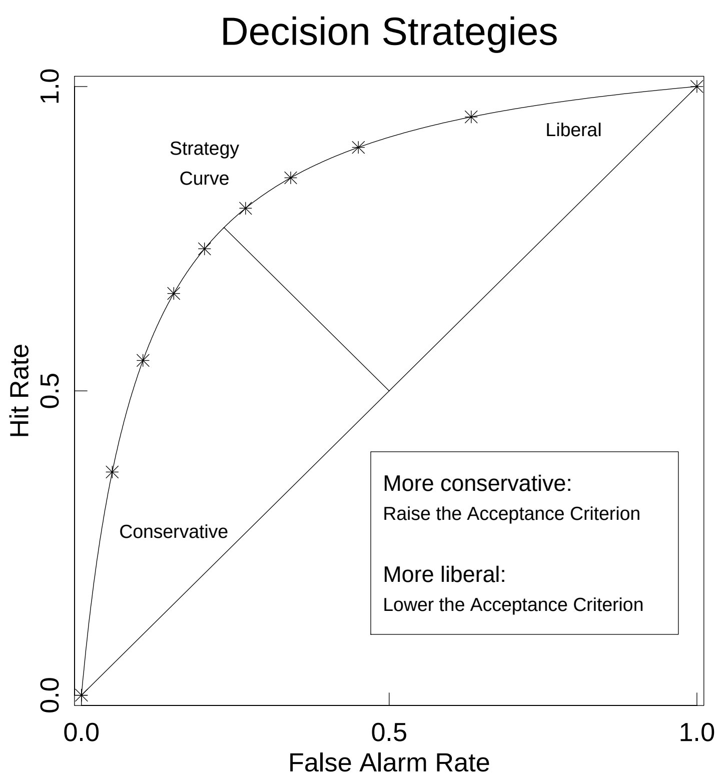


Figure 18 *ROC Curve for Two-Choice Decision Environments (e.g. making Yes-No decisions for object classification and pattern recognition). ROC curves reveal trade-offs between error rates.*

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 characterised by means μ_1 and μ_2 and standard deviations σ_1 and σ_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 factor is the relation between within-class variability and between-class variability. Pattern recognition can be performed reliably only when the between-class variability is larger than the within-class variability.

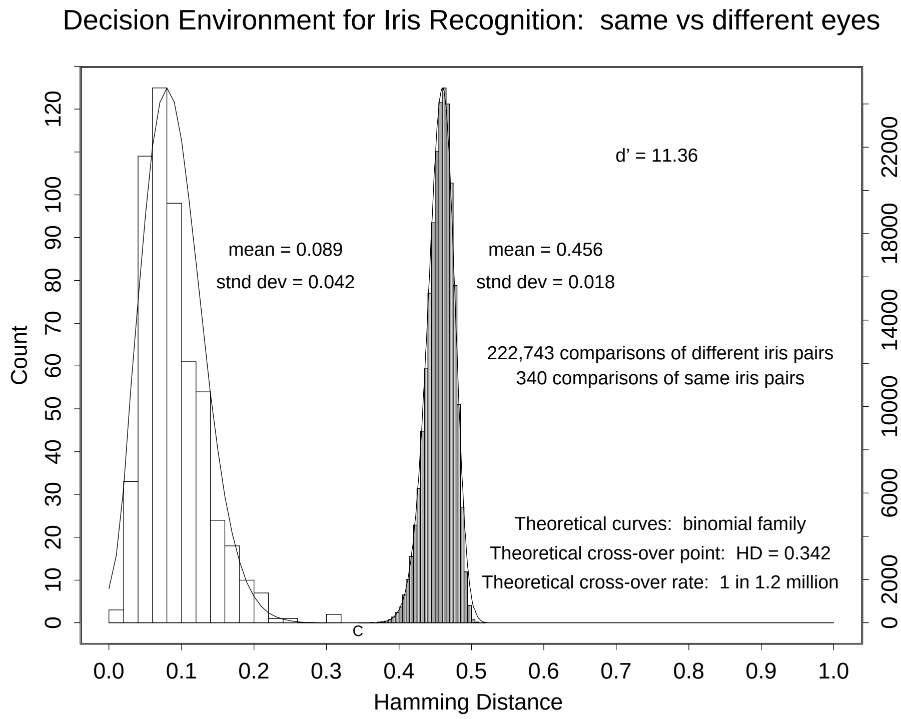
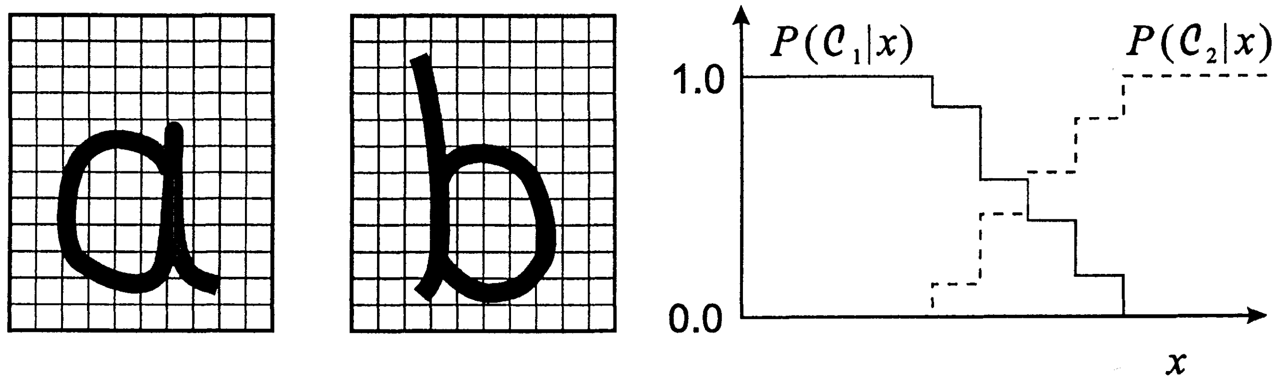


Figure 19 *A powerful decision environment with* $d' = 11.36$, *illustrating how well-separated same/different distributions lead to highly decidable classification and pattern recognition.*

10.4 Bayesian pattern classifiers

Consider a two-class pattern classification problem, such as optical character recognition on the space of just two letters, a and b. We compute some set of features x from the image data (for now we don't care what those features are), and we wish to build a Bayesian classifier that will assign a given pattern to one of two classes, C_1 or C_2, corresponding to the two letter instances.



Whatever the extracted features x are (maybe as simple as the height/width ratio), after collecting these measurements from a large number of samples of letters a and b, we can plot a histogram of how these measurements are distributed for each of the two classes. In general, these histograms will overlap, as illustrated above right. A particular sample of the value of x might come from either class $C_1 \equiv a$ or $C_2 \equiv b$; but the further to the left it is, clearly the more likely it is to have come from class C_1, other things being equal.

What do we mean by "other things being equal?" Suppose that instances of class C_2 are 100 times more frequent (more probable) than class C_1. Would we then still say that, given a slightly smallish sampled value x as indicated above, the letter class is more likely to have been C_1 than C_2?

No. Now we need to become Bayesians and take into account baseline rates. Define the prior probabilities $P(C_1)$ and $P(C_2)$ as their relative proportions (summing to 1). If we had to guess which character had appeared without our even seeing it, we would always just guess the one with the higher prior probability. Thus since in fact an 'a' is about 4 times more frequent than a 'b' in English, and these are the only two cases in this two-class inference problem, we would set $P(a) = 0.8$ and $P(b) = 0.2$.

For each class separately, we can measure how likely any particular feature sample value x will be, by empirical observation of instances from each class. This gives us $P(x|C_1)$ and $P(x|C_2)$.

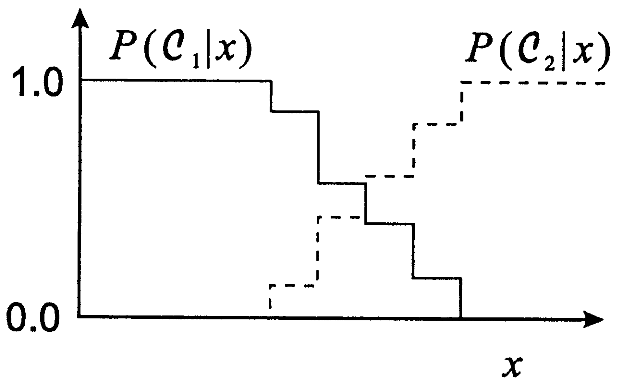
Finally, we need to know the unconditional probability $P(x)$ of any measurement value x. We can calculate this by the probability "sum rule:"

$$P(x) = \sum_{k=1}^{2} P(x|C_k)P(C_k)$$

Now we have everything we need to apply Bayes' Rule to calculate the likelihood of either class membership, given some observation x, factoring in the prior probabilities $P(C_k)$, the unconditional probability $P(x)$ of the observed data, and the likelihood of the data given either of the classes, $P(x|C_k)$. The likelihood of class C_k given the data x, is the posterior probability $P(C_k|x)$:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)} \tag{17}$$

Thus Bayes' Rule gives us a principled, formal way to perform pattern classifications on the basis of the available data and our knowledge of class baseline rates, and how likely the data would be for each of the classes. We may now plot the likelihoods of each of the classes, as a function of the data x:



We minimise the probability of misclassification if we assign each new input x to the class with the highest posterior probability. Assign x to class C_k if:

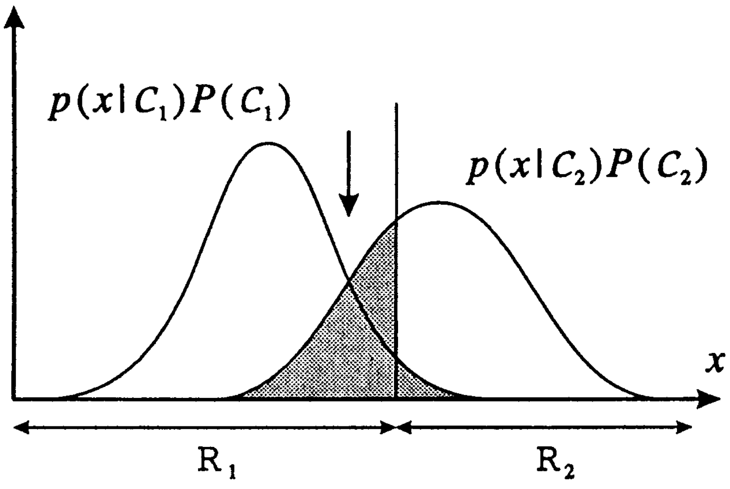
$$P(C_k|x) > P(C_j|x) \quad \forall j \neq k$$

Since the denominator in Bayes' Rule (equation 17) is independent of C_k, we can rewrite this *minimum misclassification criterion* simply as:

$$P(x|C_k)P(C_k) > P(x|C_j)P(C_j) \qquad \forall j \neq k$$

If we now plot the quantities in this inequality relation as a function of x, we can see that the minimum misclassification criterion amounts to imposing a decision boundary where the two curves cross each other (arrow):

Because the costs of the two different types of errors are not always equal, as illustrated earlier in the medical example of the biopsy, we may not necessarily



want to place our decision criterion at the point where the two curves cross, even though that would minimise the total error. If the decision boundary that we choose is as indicated by the vertical line above, then the total error is equal to the total shaded area. Let R_1 and R_2 be the regions of x on either side of our decision boundary. Then the total probability of error is:

$$\begin{aligned} P(error) &= P(x \in R_2, C_1) + P(x \in R_1, C_2) \\ &= P(x \in R_2|C_1)P(C_1) + P(x \in R_1|C_2)P(C_2) \\ &= \int_{R_2} P(x|C_1)P(C_1)dx + \int_{R_1} P(x|C_2)P(C_2)dx \end{aligned}$$

Thus the total shaded area is the total probability of error, and obviously we would minimise this (if that were our goal) by putting the decision boundary at the arrow where the two curves cross.

10.5 Discriminant functions and decision boundaries

If some set of functions $y_k(x)$ of the data x are constructed, one function for each class C_k, such that classification decisions are made by assigning an observation x to class C_k if

$$y_k(x) > y_j(x) \quad \forall j \neq k,$$

those functions $y_k(x)$ are called discriminant functions. The decision boundaries between data regions R_j and R_k are defined by those loci in the (normally multi-dimensional) data x at which $y_k(x) = y_j(x)$. A natural choice for discriminant functions would be the posterior probabilities:

$$y_k(x) = P(C_k|x)$$

Equivalently since the denominator $P(x)$ in Bayes' Rule is independent of k, we could choose

$$y_k(x) = P(x|C_k)P(C_k)$$

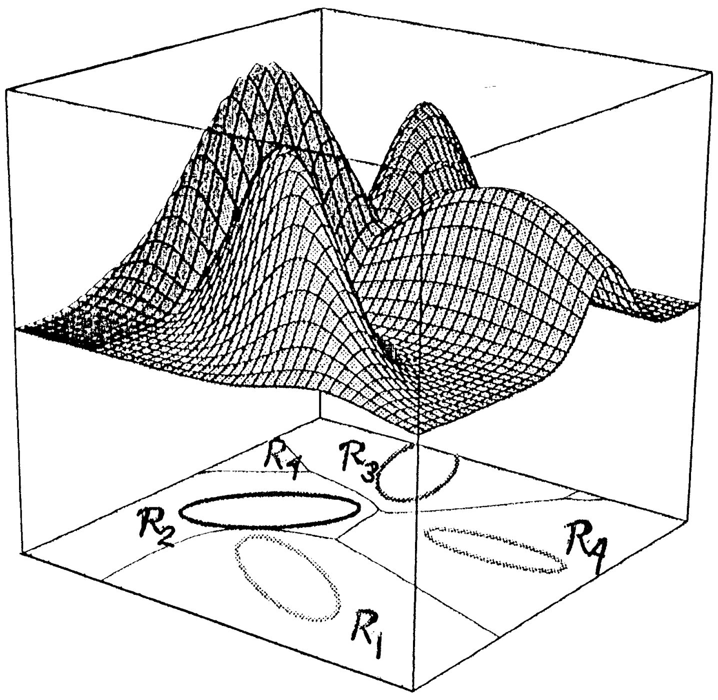


Figure 20

or any monotonic function of this, since the decision boundaries would remain the same. Figure 20 illustrates how in even just the case of two-dimensional data, the decision boundaries separating four Gaussian densities (corresponding to four classes) can be rather complex.

11 Learning and statistical methods in vision. Optical character recognition and Content based image retrieval.

The last lecture highlighted the importance of statistics, probabilities, and inference in solving computer vision tasks. In this lecture we will briefly discuss the important role that machine learning plays in modern vision systems. Machine learning and statistics are vast fields of research, so we will confine ourselves to looking at a very few examples of their use in computer vision. The next lecture will highlight further applications of these topics to the detection and recognition of human faces.

11.1 Trends in machine learning for computer vision

As mentioned in the last lecture, computer vision has seen a very strong trend towards using Bayesian techniques for inference. Classifiers based on this approach can have many parameters (conditional probabilities and independence assumptions between random variables), and there are many techniques for training factorised graphical probabilistic models, known as Bayesian networks or belief networks, from labelled data. Such methods provide a natural tool for dealing with the most common problems of engineering and science, namely **complexity** and **uncertainty**.
Probabilistic graphical models incorporate prior information about conditional independences amongst a set of variables corresponding to observations and hidden causes that are to be inferred using inference techniques. Dynamic Bayesian networks embody a stochastic state that is dynamically adapted to arrive at the most likely model ("belief") of the world, i.e. the conclusion that is best supported by the available data. Recognition can be posed as a joint inference problem relying on the integration of multiple (weak) cues to disambiguate and combine evidence in the most suitable context as defined by the top level model structure.
Another popular family of machine learning techniques in computer vision are *support vector machines* (SVM, introduced by Vladimir Vapnik and others in the 1990s). These methods are founded on the concept of structural risk minimisation as a method for learning a parameterised approximation of a target function by minimising an empirical loss criterion subject to smoothness or complexity constraints. The learned function is represented in terms of kernel basis functions, which are viewed as computing dot products between input vectors in an induced feature space. An appropriate choice of the kernel mapping allows even highly complicated decision problems to be represented by means of linear classification boundaries (hyperplanes) in the feature space. Minimising the structural risk can then be reduced to solving geometric constraints to find boundaries that maximise the margin between kernel-mapped

input vectors that are assigned different class labels. The representation of the boundaries through a small set of so-called support vectors allows very efficient classifiers to be learned from data.
Modern machine learning emphasises how older frameworks such as *artificial neural networks* can be formally analysed and generalised using the Bayesian approach and the formalisms of computational (statistical) learning theory. Another important class of methods are *unsupervised* techniques, which essentially self-organise or cluster unlabelled image data or derived features based on inherent statistical properties. The next lecture will give an example of this in the form of principal components analysis (PCA) which underlies the "Eigenfaces" approach to face recognition.

11.2 Discriminative and generative methods

The cornucopia of machine learning and data mining techniques that have become prevalent in computer vision can at times seem confusing or rather ad hoc. One way of categorising this plethora of algorithms and approaches is on the basis of whether they are primarily *generative* or *discriminative*:

- Discriminative methods learn a function $y_k(x)$ which maps input features x to class labels C_k (see section 10.5), something that can also be done probabilistically according to the posterior probabilities $y_k(x) = P(C_k|x)$. Examples include artificial neural networks, support vector machines, boosting methods, and linear discriminant analysis.

- Generative methods learn a generative likelihood model $P(x|C_k)$ which can then be used for classification using Bayes' rule. Generative models have predictive power as they allow one to generate samples from the joint distribution $P(x, C_k)$, and they are therefore popular for tasks such as the analysis and synthesis of facial expressions. Examples include probabilistic mixture models, most types of Bayesian networks, active appearance models, Hidden Markov models, and Markov random fields.

Generative models often generalise well and may therefore require less training data, but the models themselves may become more complex than is required for classification, especially for larger numbers of classes. Constructing such a model often requires specific domain expertise (e.g. for the design of a Bayesian network). On specific (supervised) learning tasks, discriminative methods usually perform better and are more efficient, but the training data needs to be large enough to span the expected modes of variation in the data.

11.3 Optical character recognition (OCR); Convolutional neural networks

OCR systems have been developed for numerous applications including postal and bank cheque routing, book digitisation, automated number plate recognition, text-to-speech synthesis for the blind, and handwriting recognition for portable device interfaces. Modern approaches make heavy use of machine learning to allow recognition of multiple fonts and to cope with distortions, noise, and variations in size, slant, and line thickness.

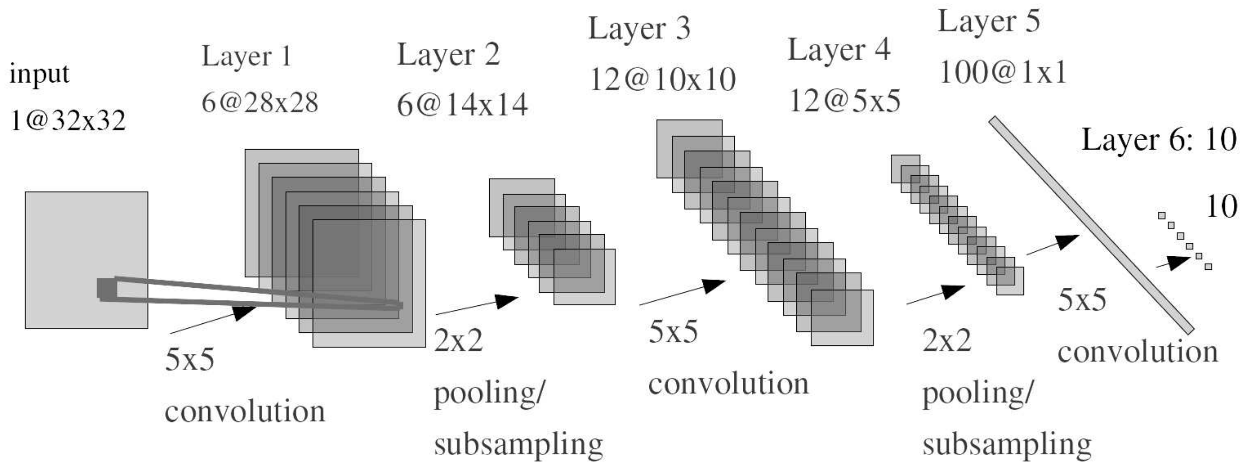


Figure 21 *Example of a convolutional neural net used for recognising handwritten digits.*

One of the most effective approaches to OCR is the convolutional neural network (conv. net). This type of artificial neural network was introduced by LeCun et al. in the 1990s and has since been applied to a range of tasks in object recognition and robotics. Unlike most machine learning approaches to computer vision, which typically rely on a manually selected set of features, conv. nets are applied directly to image data and learn their own feature representation.

The network shown in figure 21 takes a 32x32 pixel image as its input. The first stage of the network is a *convolutional layer* consisting of 6 feature maps. The neurons in each feature map have 25 adaptable weights corresponding to the elements of a 5x5 kernel which is convolved with the input image, plus an adaptable bias weight. Each feature map therefore has 28x28 $(32-5+1=28)$ neurons, all of which share the same 26 weights. In this way, each of the 6 feature maps can be trained to extract a particular position independent visual feature. As with other types of feed-forward neural network, the outputs o_{ij} of each first layer neuron i are the result of applying an activation function f_{act} (often the hyperbolic tangent, $tanh$) to the sum of its inputs (pixels in the input image I) multiplied by each of its weights w_{mn} after adding an additional bias term w_0:

$$o_{ij} = f_{act}(w_0 + \sum_m \sum_n w_{mn} I_{i-m,j-n})$$

(note how the double summation is equivalent to the 2D discrete convolution operation shown on page 23). The use of convolutional layers with shared weights was inspired by receptive field profiles in the retina (see section 2.3). Shifting the input image results in a corresponding shift in the output of the feature maps.

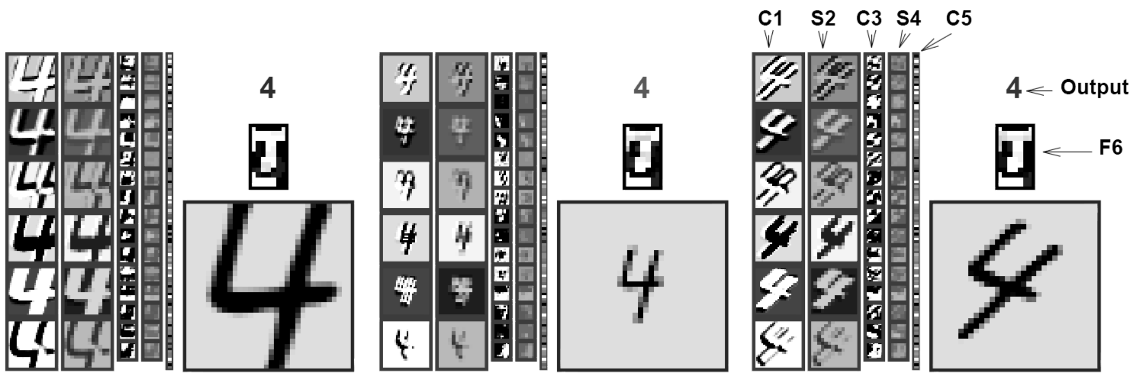


Figure 22 *Examples of a convolutional neural net applied to images of handwritten digits. The small images show the outputs of different convolutional (C) and subsampling (S) feature maps at different layers of the network in response to the input image.*

To allow combinations of visual feature responses to be processed in a more position independent way, the output of the first layer is first sub-sampled by a factor of 2 in each spatial dimension and the result is fed to another convolutional layer (layer 3 in figure 21). This layer uses a set of 12 feature maps with 5x5 kernels. The outputs of these maps are obtained by treating the outputs of the preceding layer as input images and convolving them with the kernel. Thus each neuron in the third layer obtains inputs from multiple outputs of the subsampled feature maps of layer 2 (generally all of them, although in some architectures different subsets of layer 2 maps may be convolved with different layer 3 kernels).
After another stage of subsampling, we are left with 5x5 feature maps. A final stage of convolution (layer 5) with 5x5 kernels produces single outputs ($5 - 5 + 1 = 1$). Layer 5 contains 100 such neurons. Finally there are 10 outputs corresponding to the digits 0-9, and the 10 neurons of the final layer are fully connected to each of the preceding 100 neuron outputs. Conv. nets often contain at least one fully connected layer which sometimes uses a different kind of activation function (such as a Gaussian) to implement a sort of associative pattern memory. Conv. nets are trained using the standard backpropagation algorithm, although second-order derivative approximations are also some-

times used ("stochastic diagonal Levenberg-Marquardt"). The training set may contain 10s or 100s of thousands of examples of each character (differing in style, boldness, slant, size, and with additive noise or shading to produce robust classifiers) with the corresponding target output set to +1 and all other outputs set to −1. Figure 22 shows examples of different instances of the digit 4 being recognised by a convolutional neural network. Further examples (including animations) can be found at http://yann.lecun.com/exdb/lenet/.

11.4 Content based image retrieval (CBIR)

Although images and video comprise an ever growing bulk of the world's digital content, most information retrieval systems rely entirely on textual metadata such as captions, annotations, and tags. Metadata based multimedia retrieval effectively treats images as "black boxes" since all indexing and search is based on the labels associated with a given image rather than the image itself. Furthermore, manual image annotation is an expensive process which is prone to errors, inconsistencies, ambiguity, lack of context, and both over- and under-keywording. A set of textual annotations effectively becomes immutable (not amenable to modification or re-interpretation) and is tied to the idiosyncrasies of a particular natural language.
Consequently there is great scope for systems that are able to perform image search on the basis of an automated analysis of the actual content of images. However, most systems for content-based image retrieval (CBIR) generally only provide search using low-level image features such as colour or texture statistics. CBIR has suffered from too much emphasis being placed on a system view of the retrieval process in terms of image processing, feature extraction, content representation, data storage, matching, etc.. Indeed, one criticism one can generally level at CBIR systems is the extent to which they require the user to model the notions of content representation and similarity employed by the system, rather than vice versa. Most CBIR systems offer one or more of the following query mechanisms:

- *Feature range or predicate*: Here the user can set target ranges or thresholds for certain (typically low-level) attributes such as colour, shape, or texture. This kind of interface requires a certain amount of user sophistication and patience and is ill-suited to retrieval based on higher-level concepts.

- *Template, region selection, or sketch*: The user can draw (sometimes literally) the system's attention to particular image aspects such as the spatial composition of desired content in terms of particular regions or a set of pre-defined templates. Clearly this process becomes cumbersome for complex queries.

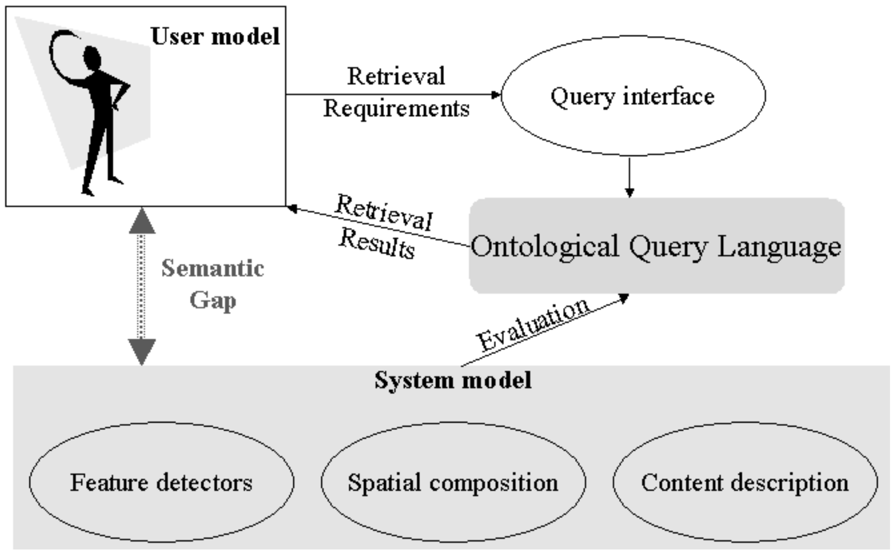


Figure 23 *CBIR systems are particularly prone to the semantic gap between human and computer capabilities for interpreting image content. One approach that has been proposed for bridging this gap is the use of an ontological query language.*

- *Query-by-example*: Finding suitable example images can be a challenge and may require the user to manually search for such images before being able to query the automated system. It is also difficult for the system to ascertain which aspects make a given image relevant and how similarity should be assessed. However, modern *image similarity* search systems are becoming popular for tasks such as online shopping, stock photography search, image clustering, and detection of copyright infringement.
- *Query language or concept*: Some CBIR query languages are based on SQL or Boolean query constructs. Knowledge-based approaches utilising description logics or semantic networks have been proposed as a means of better representing semantic concepts but tend to entail somewhat cumbersome query interfaces.

One of the problems of CBIR is the fact that visual information is inherently ambiguous and semantically impoverished. There consequently exists a wide *semantic gap* between human interpretations of visual information and the recognition capabilities of computer vision systems. CBIR has the added challenge of inferring user retrieval requirements from the query, and efficiently determining the more relevant images from a collection of potentially billions of images. A recent approach to bridging this gap is to make use of an *ontological query language*, combined with a set of advanced automated image analysis and classification modules.
Ontology is the theory of objects in terms of the criteria which allow one to distinguish between different types of objects and their relationships, depen-

$$
\begin{array}{lll}
G: \{ & & \\
S & \rightarrow & R \\
R & \rightarrow & \mathit{modifier}?\ (\mathit{metacategory} \mid SB \mid BR) \\
 & & \mid \mathit{not}?\ R\,(CB\ R)? \\
BR & \rightarrow & SB\ \mathit{binaryrelation}\ SB \\
SB & \rightarrow & (CS \mid PS) + LS* \\
CS & \rightarrow & \mathit{visualcategory} \mid \mathit{semanticcategory} \mid \\
 & & \mathit{not}?\ CS\,(CB\ CS)? \\
LS & \rightarrow & \mathit{location} \mid \mathit{not}?\ LS\,(CB\ LS)? \\
PS & \rightarrow & \mathit{shapedescriptor} \mid \mathit{colourdescriptor} \mid \\
 & & \mathit{sizedescriptor} \mid \mathit{not}?\ PS\,(CB\ PS)? \\
CB & \rightarrow & \mathit{and} \mid \mathit{or} \mid \mathit{xor}; \\
\} & &
\end{array}
$$

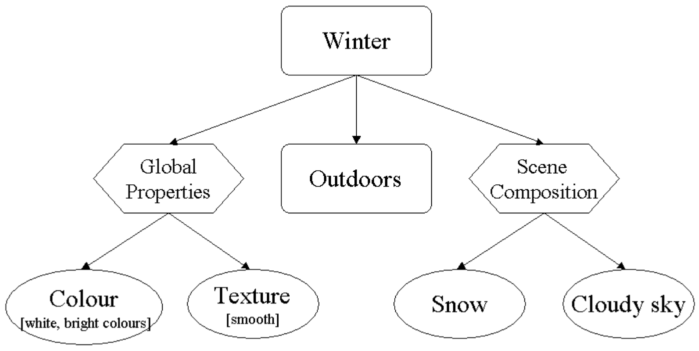


Figure 24 *Left: Example of a grammar defining the interface to an ontological query language. Right: Simplified Bayesian network for the scene descriptor "winter".*

dencies, and properties. Ontologies encode the relational structure of concepts which one can use to describe and reason about aspects of the world. This makes them eminently suitable for many problems in computer vision which require prior knowledge to be modelled and utilised in both a descriptive and prescriptive capacity.

Image retrieval can then be carried out by processing sentences in a visual language defined over the ontology. User queries are parsed into a canonical representation which is then linked to automatically recognised image content in accordance with the retrieval need expressed by the query. The underlying ontology encompasses relational information about concepts and attributes pertaining to automatically recognised image content, as well as knowledge about the structure and meaning of natural language queries expressed in English. The relevance of each image in a collection with respect to a given user query is assessed probabilistically while taking into account both the reliability and salience (as it pertains to the query) of all information available for that image. Figure 24 gives an example of a simplified grammar for such a query language and an example of how a Bayesian network can be used to implement part of the image content inferences used by the indexing and retrieval system.

Query sentences are typically short (e.g. "two people at the beach during sunset") and need only represent those aspects of the target image(s) which the user is trying to retrieve and which distinguish such images from others in the dataset. The user is therefore not required to translate a description of an envisaged target image into the language but merely (and crucially) to express desired properties that are to hold for the retrieved images. Hence

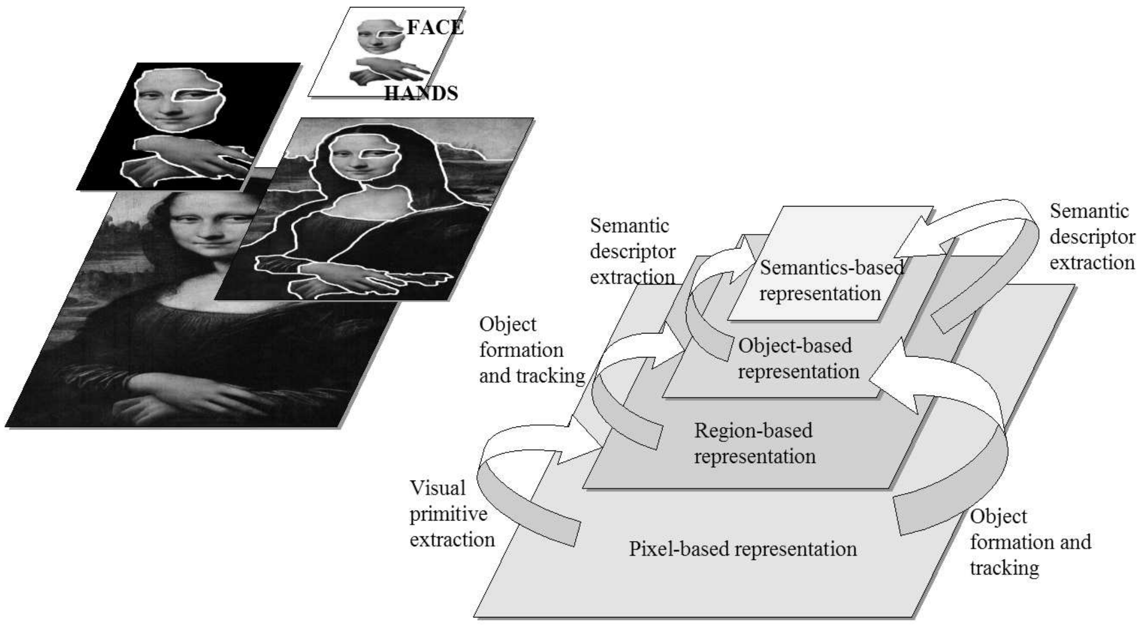

FACE
HANDS
Semantic descriptor extraction
Semantics-based representation
Semantic descriptor extraction
Object formation and tracking
Object-based representation
Region-based representation
Visual primitive extraction
Pixel-based representation
Object formation and tracking

Figure 25 *Hierarchical recognition and representation of visual information at different levels and example illustrating these concepts as applied to an image of the "Mona Lisa" (original copyright: Musee de Louvre).*

even a fairly short query sentence can suffice to select a small subset of desired images from a vast collection. This simple idea is the reason why text retrieval on the internet is so successful: the less frequently a particular constellation of keywords appears across the entire document set, the more valuable it is as a means of discriminating relevant from non-relevant content.
As illustrated in figure 25, the system can make use of an ontology of image content representations extracted using a range of techniques such as:

- Image segmentation: In order to identify salient parts of the image corresponding to objects or object parts, the image is automatically segmented into a covering set of non-overlapping regions and sets of properties such as size, colour, shape, and texture are computed for each region. The number of segmented regions depends on image size and visual complexity, but has the desirable property that most of the image area is usually contained within a few dozen regions which closely correspond to the salient features of the picture.

- Region classification: SVMs and other classification methods are used to recognise material and environmental categories, such as "grass", "sky", "wood", "water". This may be regarded as an intermediate level semantic representation which serves as the basis for subsequent stages of visual inference and composite object recognition.

- Scene classification: A second stage of Bayesian network classifiers is applied to analyse image content at a higher scene level. Examples of scene categories include “indoor”, “beach”, “sunset”, “nighttime”, “autumn”, etc..

- Object detection and recognition: The image analysis also features detectors for common objects such as cars and buildings. Human faces are automatically detected and classified according to personal attributes such as gender, age, and facial expression.

12 Face detection, recognition, and interpretation

Louis Léopold Boilly, Reunion de Têtes Diverses

The goal of detecting faces and recognising 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 generally 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 recognise faces, but also to interpret and classify their expressions.)

12.1 Issues in detecting, recognising, and interpreting faces

As usual, this domain of computer vision raises questions such as:

1. What is the best representation to use for faces?
2. Must this be treated as a 3D (object-based) or 2D (image-based) problem?
3. How can invariances to size (hence distance), location, pose, 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, in order to reliably detect the presence of a face?
5. What are the *particular* features that we can rely upon to recognise the identity of any given face?
6. What is the best way to handle "integration of evidence," and incomplete information, and to make decisions under uncertainty?
7. How can we handle the transformations that can occur in a given person's face, either through natural, or unnatural means?

12.2 The fundamental problem of pattern recognition

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. 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 very disappointing by contrast, with error rates approaching 50%.

The array of images in figure 26 show how dramatic are the effects of even

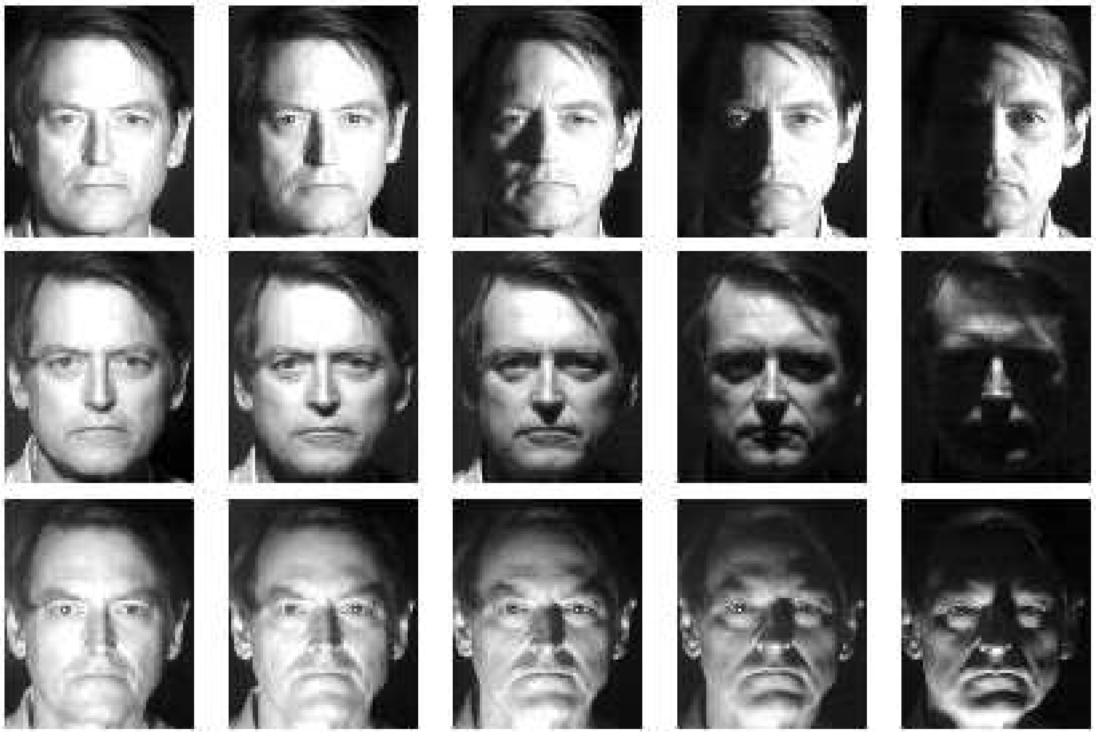


Figure 26

only a change in illumination *direction.* Facial expression remains exactly the same. Going across the columns from left to right, the illumination changes from frontal to side; and going down the rows, it changes in elevation. If you compare the 3 images in the last column on the right, it seems almost inconceivable that any means could be found to represent these as images of the same person.

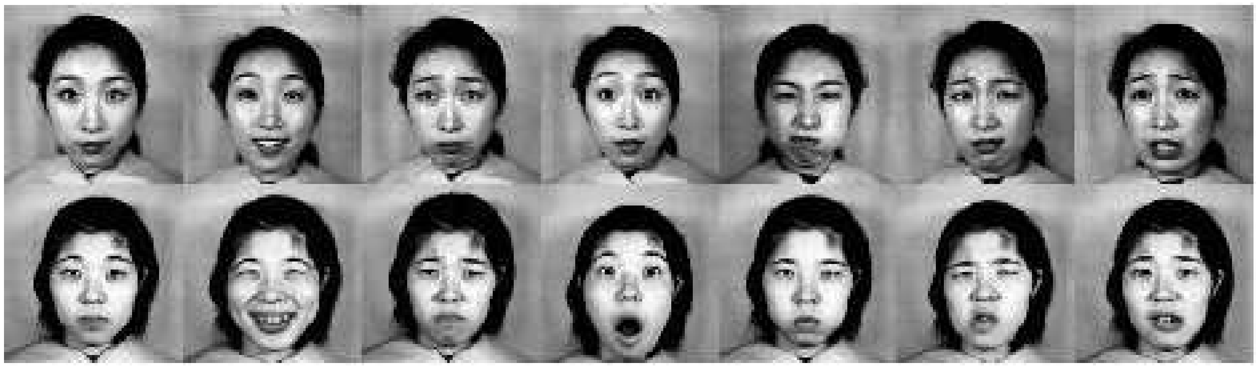


Figure 27

Earlier (see figure 1) we saw how dramatically a change in pose angle affects image appearance, even though the expression and illumination remained the same. Appearance-based algorithms for face recognition still tend to judge different faces in the same pose as more similar than identical faces in different poses. Finally, the images in figure 27 show how much a given person's face (in each row) can change when she is using it socially as an organ of expression.

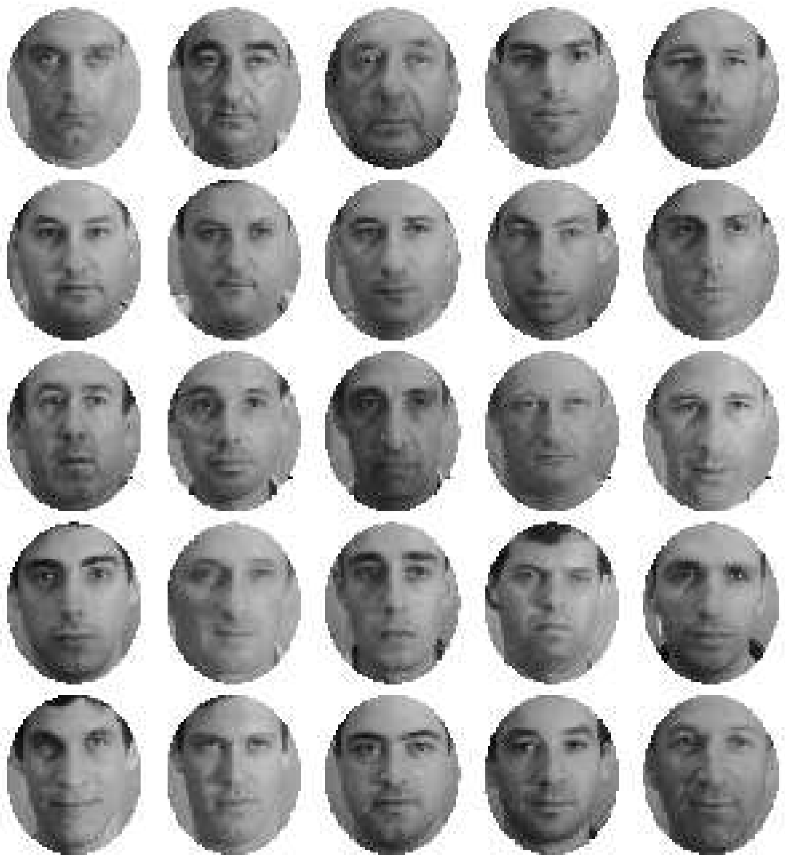


Figure 28

For comparison now, when we examine images of *different* faces seen under fixed illumination and with neutral expressions, their (between-class) variability seems <u>tiny</u> compared to the same-person (within-class) variabilities above associated with changes either in illumination or in expression (figure 28).
When there is variability across two or more dimensions (let us say both face identity and facial expression, as in figure 29), then discriminability might benefit from variability *within* a class of the other dimension, but not from variability *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 amongst different faces. The consequences of within-class and between-class variability, for single dimensions and across them, are noted in the following table:

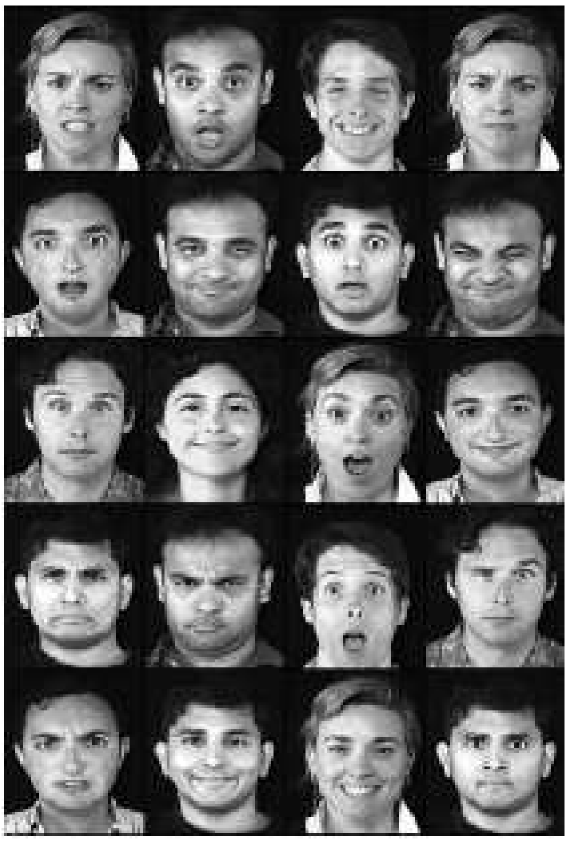


Figure 29

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

Many of these forms of variation in facial appearances were captured in the painting by Boilly, *Reunion de Têtes Diverses* (see beginning of this section). In characterising the within-class variability and the between-class variability of faces, it is clear that (at least over 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 ("epigenetic features"). Persons who are genetically identical would share all their genotypic features, such as gender, blood group, race, and DNA sequence, whereas epigenetic features can be shared among different individuals only by chance, according to their associated probability distributions.

One source of evidence about the genetic/epigenetic ratio of facial variation arises from genetically 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 determination of facial structure.

A further, but secondary, indicator of the genetic determination 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 Kennedy in adulthood.

Interestingly, a major part of the computational load of the brain is concerned with "social computation," a large part of which involves identifying and interpreting faces. It is generally accepted among ethologists and neuroscientists that the main evolutionary pressures that led to the large brains of primates were not "engineering oriented" pressures such as learning to use tools, but rather the demands of sexual competition. Included in those task demands are: seduction; betrayal; assessing power hierarchies and your own place within them ["who is the alpha male here?"]; manipulation of others' intentions and desires; and interpreting those within others [i.e. the "other minds" problem].

12.3 Face detection

Paradoxically, face detection is a harder problem than face recognition, and the performance rates of algorithms are often poorer. (This seems paradoxical since detection must precede recognition; but recognition performance is measured only with images already containing faces.) To improve performance, many face detectors make use of ancillary cues such as the presence of skin.

The rather special hue composition of human skin is matched by few other types of surfaces. (Racial differences correspond only to variations in saturation, due to differential melanin density, but not to large differences in hue.) Approaches to face detection often use generic templates, spanning multiple scales (for faces at different distances, hence sizes) and poses. This sliding-window approach to detection suffers from the drawback that the template may need to be compared with the image at many different positions and scales. Starting with a detector size of 20x20 pixels and evaluating it at all possible offsets in a 400x400 image would require $(400 - 20 + 1)^2 = 145161$ evaluations, just for a single scale of analysis! Clearly such a detector would have to be very efficient and have an extremely low false alarm rate to give reasonable performance. In practice one would shift the detector window by more than one pixel at a time depending on the current window size[1], and the scale would be increased by some constant (say 20%) at each iteration over the image, but the number of evaluations will still be about 10^5 per image.

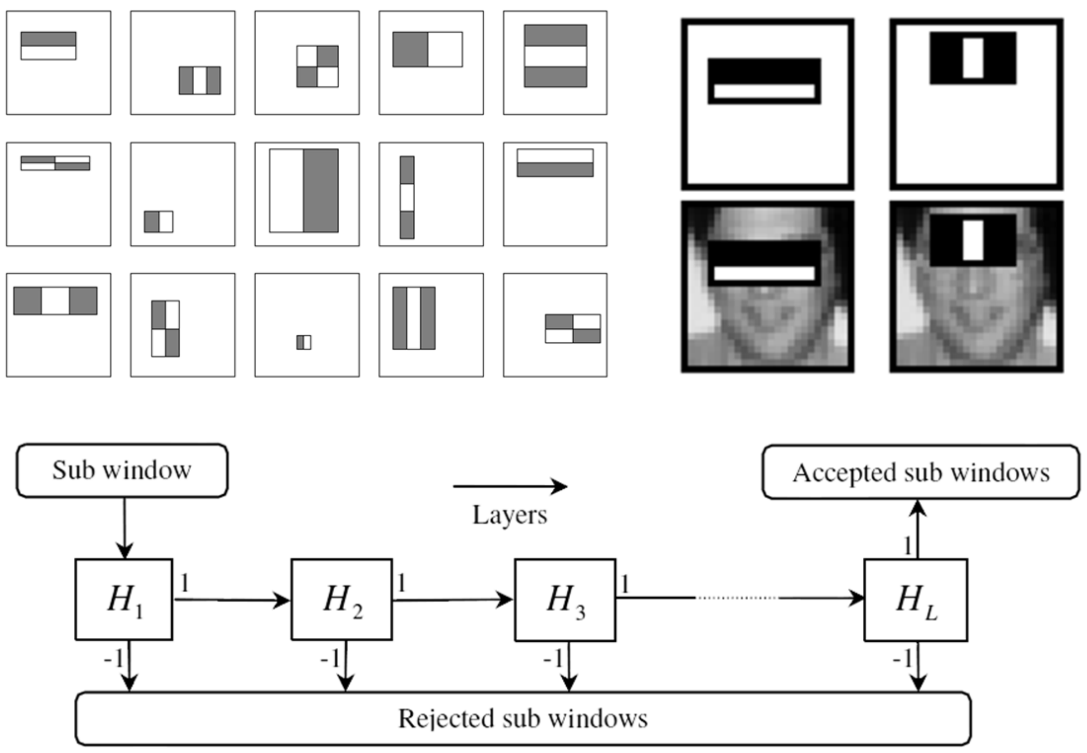


Figure 30 *Viola-Jones face detection method. Top left: examples of rectangle features. Top right: example of how even very simple features can be used to detect rudimentary aspects of faces such as eyes and nose. Bottom: illustration of the rejection-cascade architecture.*

Modern approaches to face detection make use of a number of image pro-

[1]A usable detector would also incorporate some rules for disambiguating multiple neighbouring detections resulting from overlapping candidate image regions.

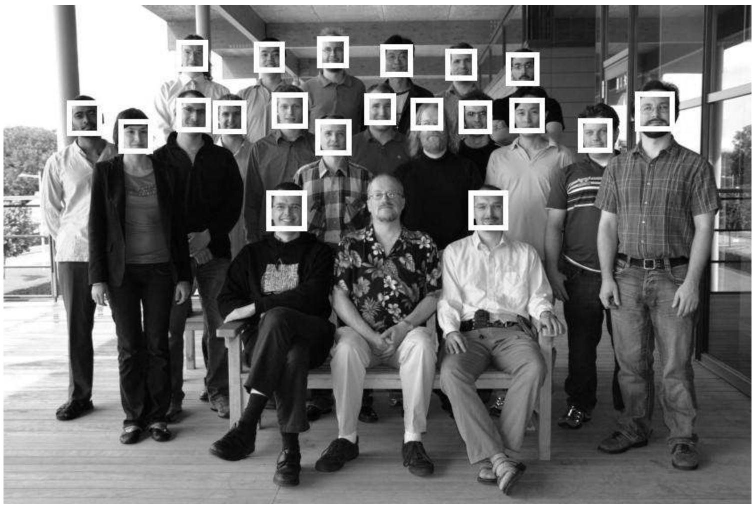
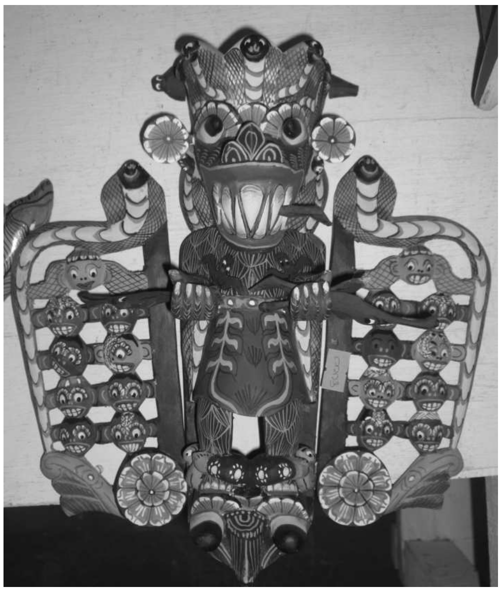



Figure 31 *Examples of face detections using the Viola-Jones approach.*

cessing and machine learning techniques to deal with these challenges. The currently most popular method is due to Viola and Jones (2004), who popularised the use of the AdaBoost ("Adaptive Boosting", formulated by Freund and Schapire) machine learning algorithm to train a cascade of feature classifiers for object detection and recognition. Boosting is a supervised machine learning framework which works by building a "strong classifier" as a combination of (potentially very simple) "weak classifiers". A Viola-Jones face detector consists of classifiers based on simple rectangular features (which can be viewed as approximating Haar wavelets) and makes use of an image representation known as the integral image (also called summed area table) to compute such features very efficiently. The resulting boosted classifier is a weighted combination of thresholded responses to a set of rectangular features that, like Haar basis functions, differ in complexity (i.e. the features may consist of 2, 3 or 4 rectangular regions), scale, position, and orientation (horizontal or vertical, though some implementations also incorporate diagonal features). Formally, a weak classifier $h_j(x)$ consists of a feature f_j, a threshold θ_j and a parity $p_j \in \pm 1$ such that

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j < p_j \theta_j \\ -1 & \text{otherwise} \end{cases}$$

and the resulting strong classifier using weights a_j is

$$h(x) = sign(\sum_j a_j h_j)$$

By combining such classifiers into a hierarchical cascade made up of increasingly complex classifiers, good detection accuracy can be achieved at relatively low false positive levels. The cascade is also very efficient, since each stage

(layer) is computationally very simple to apply to an image region and only those regions which are accepted by a given layer of the cascade ($h(x) > 0$) are passed on to the next layer for consideration. Training is done in such a way that early cascade layers have very high true accept rates (with correspondingly high false positive rates, as discussed in section 10.3) in order to quickly reject those image regions that are very unlikely to represent a face. Later stages are trained to be more discriminating and consequently have increasingly lower target false positive rates. Each stage is trained by adding rectangle features until the target detection and false positive rates are met.
A fully trained face detection cascade may have over 30 layers, yet the vast majority of candidate image regions will only be considered by the first few of these. To perform face detection, the cascade is evaluated at different scales and offsets within an image using a sliding window approach. Figure 30 illustrates this method. The overall detection rate D of a cascaded detector with N "rejection" layers is

$$D = \prod_{i=1}^{N} d_i$$

where d_i is the true accept rate of layer i, and similarly for the overall false positive rate

$$F = \prod_{i=1}^{N} f_i$$

Since we may need to consider 10^5 sub-regions in a given image, we want F to be less than 10^{-5} in order to expect fewer than one false positive detection per image. To achieve $F = 10^{-5}$ for a 30 layer cascade, each f_i would have to be about 68% ($10^{-5/30} = 10^{-1/6}$), which looks rather easier than creating a single monolithic classifier with a false alarm rate below 0.00001! However, by the same argument, a decent detection rate of $D = 0.95$ would require d_i of $.95^{1/30}$ which is about 99.83%. Clearly, as discussed in section 10, the optimal choice of the trade-off between the two error rates depends on the prior probability of any given image region containing a face (we expect far fewer than 100,000 discernable faces in an image), and the required target error rates of the learning algorithm can be determined from a data set. The two examples in figure 31 show excellent (but not perfect!) recognition performance without false positives, despite the many face-like characteristics of the right image.
One major drawback of Viola-Jones and practically all approaches to face detection is the lack of invariance to orientation (in-plane rotation) and pose (out-of-plane rotation) of faces. Real-world face detectors usually consist of multiple detectors trained for particular ranges (typically $\pm 15\,\mathrm{deg}$) of pose and orientation, and these component detectors are either applied in parallel or based on some rough prior detection and pose estimation step.

12.4 Two-dimensional (appearance-based) approaches to face recognition

One of the most prominent approaches to face recognition is the *Eigenfaces* approach (Kirby and Sirovich; Turk and Pentland). This involves performing a complete Karhunen-Loeve Transform of a large database of faces (typically more than 10,000) to extract the principal components, i.e. the 20 or so main 2D 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 often in the 90% to 95% range. However, a limitation in this method is that many of the principal components simply extract variations due to shading caused by variations in the angle of illumination! Other high-order principal components are extracting variations in the outline of the face due to small errors in size (distance) normalisation. Thus the method is essentially a 2D representation of faces, and lacks invariances to illumination or pose angle, or any real size invariance.

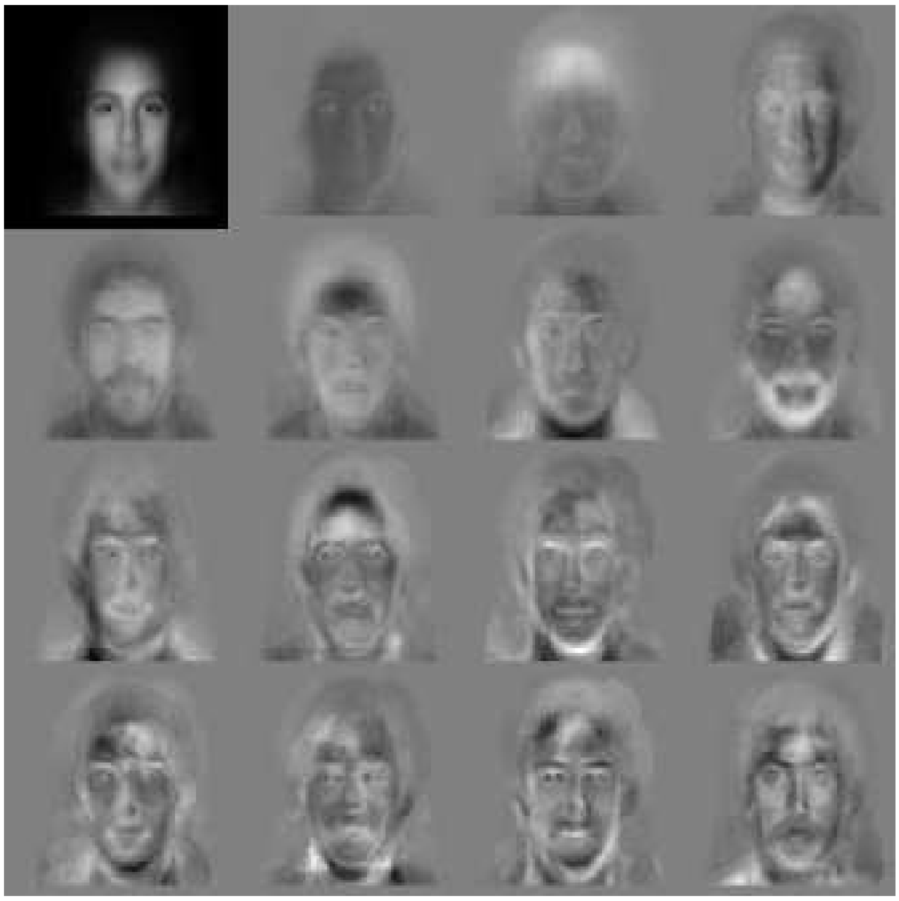


Figure 32

The Eigenfaces approach exemplifies a typical strategy in computer vision,

which is *projection to a low-dimensional subspace.* The critical variation in a pattern recognition problem is captured in some low dimensional set of basis vectors, such as the 20 most important "eigenfaces" to emerge from Principal Components Analysis (PCA) of a dataset that is regarded as representative of the problem domain. Those are then treated as basis vectors, in terms of which any face is represented by some linear combination. The weights, or coefficients, that specify each such linear combination are the eigenvalues; in effect they indicate the "relative presence" of each of the eigenfaces within any given presenting face. They are computed simply by taking the inner product of the presenting face image with each of the eigenfaces. Because the eigenfaces which emerge from the PCA are, by construction, orthogonal, this is a relatively rapid computation. The projection coefficients obtained serve also as expansion coefficients, since this specified linear combination of roughly 20 eigenfaces will superimpose into a very close approximation to the face in question.
Thus a face is effectively represented by a small set of numbers: the eigenvalues. Such a "face code" is extremely compact, and databases can be searched very rapidly since the description of each face is a simple feature vector of only 20 numbers. Figure 32 illustrates 15 eigenfaces computed from PCA as basis vectors, and their linear combination to superimpose into the face in the top left.

12.5 Wavelet approaches to face recognition

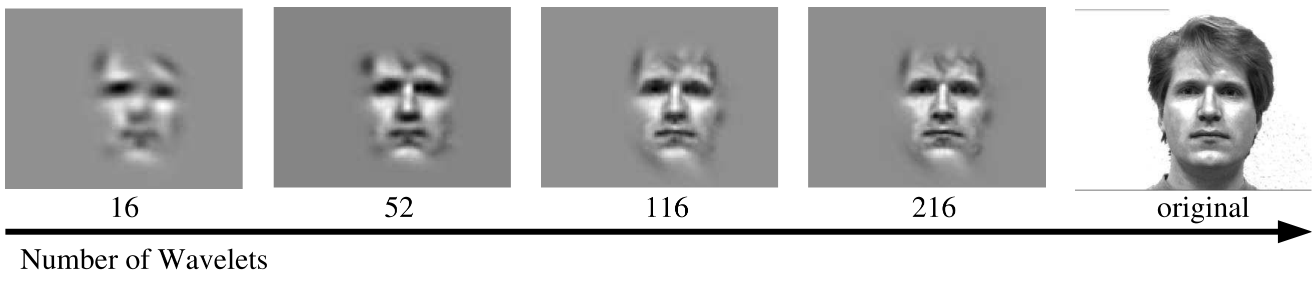
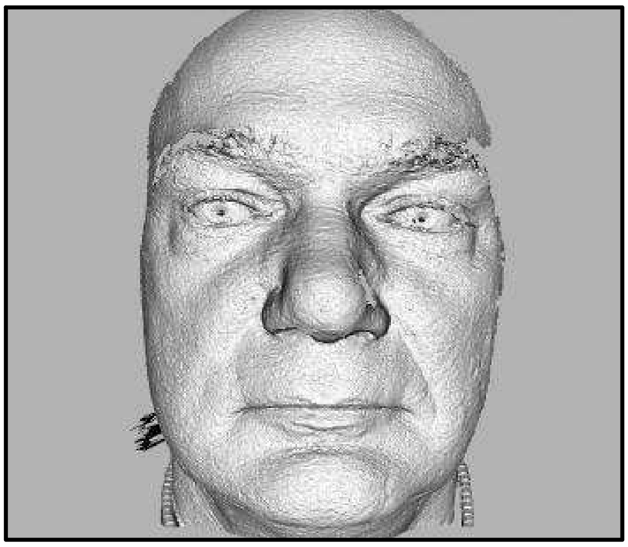


Figure 33

Recently much interest has developed in *wavelet* representations of 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 localised, they can track changes in facial expression in a local way. This approach essentially treats a face as a kind of *texture*, made up of various undulations in various positions, sizes, and orientations but without incorporating explicit models for the individual parts of faces. 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 can the entire face (see the example in figure 33).

12.6 Three-dimensional approaches to face recognition

Many current efforts in face recognition seek to model faces as three-dimensional objects, even as dynamic objects, in order to achieve invariance both to pose angle and illumination geometry. Of course, this requires solving the ill-posed problems of inferring shape from shading, interpreting albedo versus variations in Lambertian and specular surface properties, and structure from motion. Earlier we examined how difficult this problem is, and how remarkable it is that we humans seem to be so competent at it. The synthesis of vision as model-building and graphics, to perform face recognition in object-based terms, rather than appearance-based terms, is now a major focus of this field. In order to construct a 3D representation of a face (so that, for example, its appearance can be predicted at different pose angles), it is necessary to extract separately both a shape model and a texture model (texture encompasses albedo, colouration, any 2D surface details, etc).



The 3D shape model (above right) is extracted by various means, which may include laser range-finding (with millimetre resolution); stereo cameras; projection of structured light (grid patterns whose distortions reveal shape); or extrapolation from a multitude of images taken from different angles (often a 4×4 matrix). The size of the data structure can be in the gigabyte range, and significant time is required for the computation. Since the texture model is linked to coordinates on the shape model, it is possible to project the texture

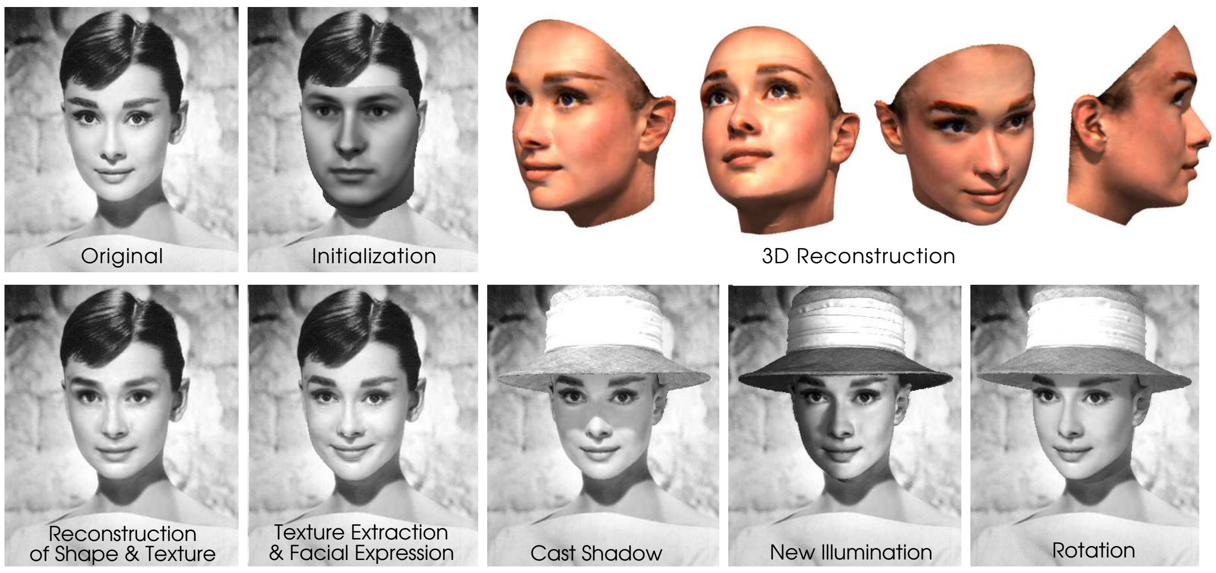


Figure 34 *A single 2D photograph (top left) can be used to morph a 3D face model after manual initialisation to build a 3D representation of the face from the photo that can be manipulated for differing pose angles, illumination geometries, and even expressions. (Blanz & Vetter)*

(tone, colour, features, etc) onto the shape and thereby generate models of the face in different poses. Clearly sensors play an important role here for extracting the shape model, but it is also possible to do this even from a single photograph if sufficiently strong Bayesian priors are also marshalled, assuming an illumination geometry and universal aspects of head and face shape.
An impressive demonstration of this process can be seen on YouTube at: http://www.youtube.com/watch?v=nice6NYb_WA . As summarised in Blanz & Vetter's paper, *Face Recognition Based on Fitting a 3D Morphable Model,* it is:

> *"...a method for face recognition across variations in pose, ranging from frontal to profile views, and across a wide range of illuminations, including cast shadows and specular reflections. To account for these variations, the algorithm simulates the process of image formation in 3D space, using computer graphics, and it estimates 3D shape and texture of faces from single images. The estimate is achieved by fitting a statistical, morphable model of 3D faces to images. The model is learned from a set of textured 3D scans of heads. Faces are represented by model parameters for 3D shape and texture."*

12.7 Approaching human performance in face recognition

Organisations such as NIST periodically run competitions for face recognition algorithms over a wide range of conditions such as: controlled/uncontrolled

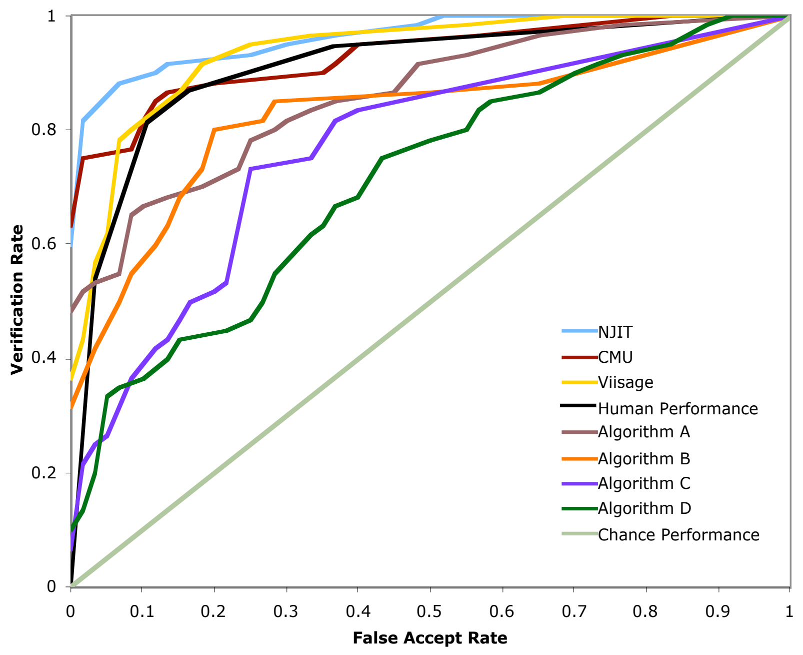


Figure 35

illumination and pose; resolution; capture interval; and 2D *versus* 3D sensors. Uncontrolled illumination and pose remain highly challenging for algorithms. Under some conditions, with high resolution ($>$ 6 megapixel) image arrays sufficient to resolve tiny details of skin texture, machine performance equals and even exceeds human performance (although some might question whether this is really "face recognition" since it needs only a few square inches of high-resolution skin texture above the eyebrows, and requires minutes to encode and match these minutiae). Figure 35 shows ROC curves reported in 2007 for several algorithms; three of them are consistently above (better than) the face recognition performance of humans (the black, fourth ROC curve).

12.8 Interpreting facial expressions

Finally, much current effort is devoted to recognising facial expressions, not only as static aspects of pictures but as dynamic sequences, unfolding in time. Motion energy models are used to extract motion signatures from specific parts of faces, and to classify these as expressions. This task also entails vision-as-inverse-graphics to construct models based upon knowledge of the human facial musculature and behaviour, while Hidden Markov Models (HMMs) capture articulated expressive state sequences. Future human-machine interaction may incorporate this interpretive aspect, known as "affective computing."