Computer Vision

Professor John Daugman

University of Cambridge

Computer Science Tripos, Part II
Lent Term 2019/20

Lecture Topics

1. **Overview. Goals of computer vision; why they are so difficult.**
2. **Pixel arrays, CCD / CMOS image sensors, image coding.**
3. **Biological visual mechanisms, from retina to visual cortex.**
4. **Mathematical operations for extracting structure from images.**
5. **Edge detection operators; gradient field; Laplacian zero-crossings.**
6. **Multi-resolution. Active Contours. Wavelets as primitives; SIFT.**
7. **Higher brain visual mechanisms; streaming; reciprocal feedback.**
8. **Texture, colour, stereo, and motion descriptors. Disambiguation.**
9. **Lambertian and specular surface properties. Reflectance maps.**
10. **Shape description. Codons; superquadrics and surface geometry.**
11. **Perceptual organisation and cognition. Vision as model-building.**
12. **Lessons from neurological trauma and deficits. Visual illusions.**
13. **Bayesian inference. Classifiers; probabilistic decision-making.**
14. **Model estimation. Machine learning and statistical methods.**
15. **Optical character recognition. Content-based image retrieval.**
16. **Face detection, face recognition, and facial interpretation.**

Aims of this course:

– to introduce the principles, models and applications of computer vision, as well as some mechanisms used in biological visual systems that might inspire design of artificial ones. At the end of the course you 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, algorithms, and inference principles;
- understand the roles of image transformations and their invariances;
- describe detection of features, edges, shapes, motion, and textures;
- describe some key aspects of how biological visual systems work;
- consider ways to try to implement biological visual strategies in computer vision, despite the enormous differences in hardware;
- be able to analyse the robustness, brittleness, generalisability, and performance of different approaches in computer vision;
- understand roles of machine learning in computer vision, including probabilistic inference, discriminative and generative methods;
- understand in depth at least one major vision application domain, such as face detection, recognition, or interpretation.

Online resources and recommended books

- CVonline: "Evolving, Distributed, Non-Proprietary, On-Line Compendium of Computer Vision" (Univ. of Edinburgh; updated in late 2019; includes many Wikipedia links): http://homepages.inf.ed.ac.uk/rbf/CVonline/
- Matlab Functions for Computer Vision and Image Processing: http://www.peterkovesi.com/matlabfns/index.html
- Annotated Computer Vision Bibliography (updated at end of 2019): http://www.visionbib.com/bibliography/contents.html
- Datasets: http://homepages.inf.ed.ac.uk/rbf/CVonline/Imagedbase.htm
- Software packages: http://homepages.inf.ed.ac.uk/rbf/CVonline/SWEnvironments.htm
- A collection of **Written Exercises** for this course (past Tripos Questions) is provided on the course website, with weekly assignments. These will be reviewed in a series of **Examples Classes** (within the lecture slots).

- Online books: http://homepages.inf.ed.ac.uk/rbf/CVonline/books.htm
- Shapiro, L. & Stockman, G. (2001). *Computer Vision*. Prentice Hall.
- Duda, R.O., Hart, P.E., & Stork, D.G. (2001) *Pattern Classification* (2nd Ed).

1. Examples of computer vision applications and goals:

- automatic face recognition, and interpretation of facial expression
- tracking of persons and objects; pose estimation; gesture recognition

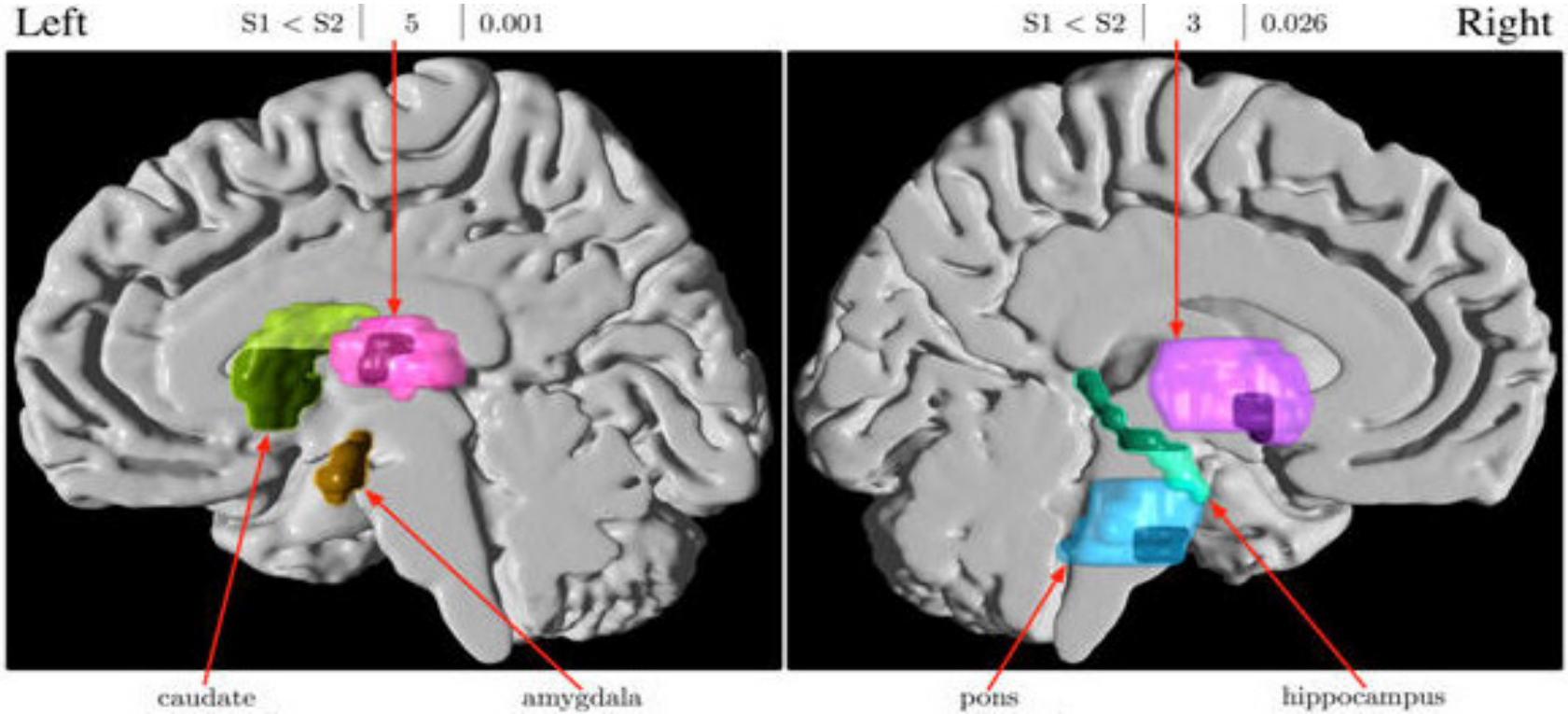
- object and pattern recognition; 3D scene reconstruction from images
- biometric-based visual determination of personal identity
- image search and content-based image retrieval; scene understanding

(some computer vision applications and goals, con't)

- vision-based autonomous robots; driverless cars
- motion estimation; collision avoidance; depth and surface inference

(some computer vision applications and goals, con't)

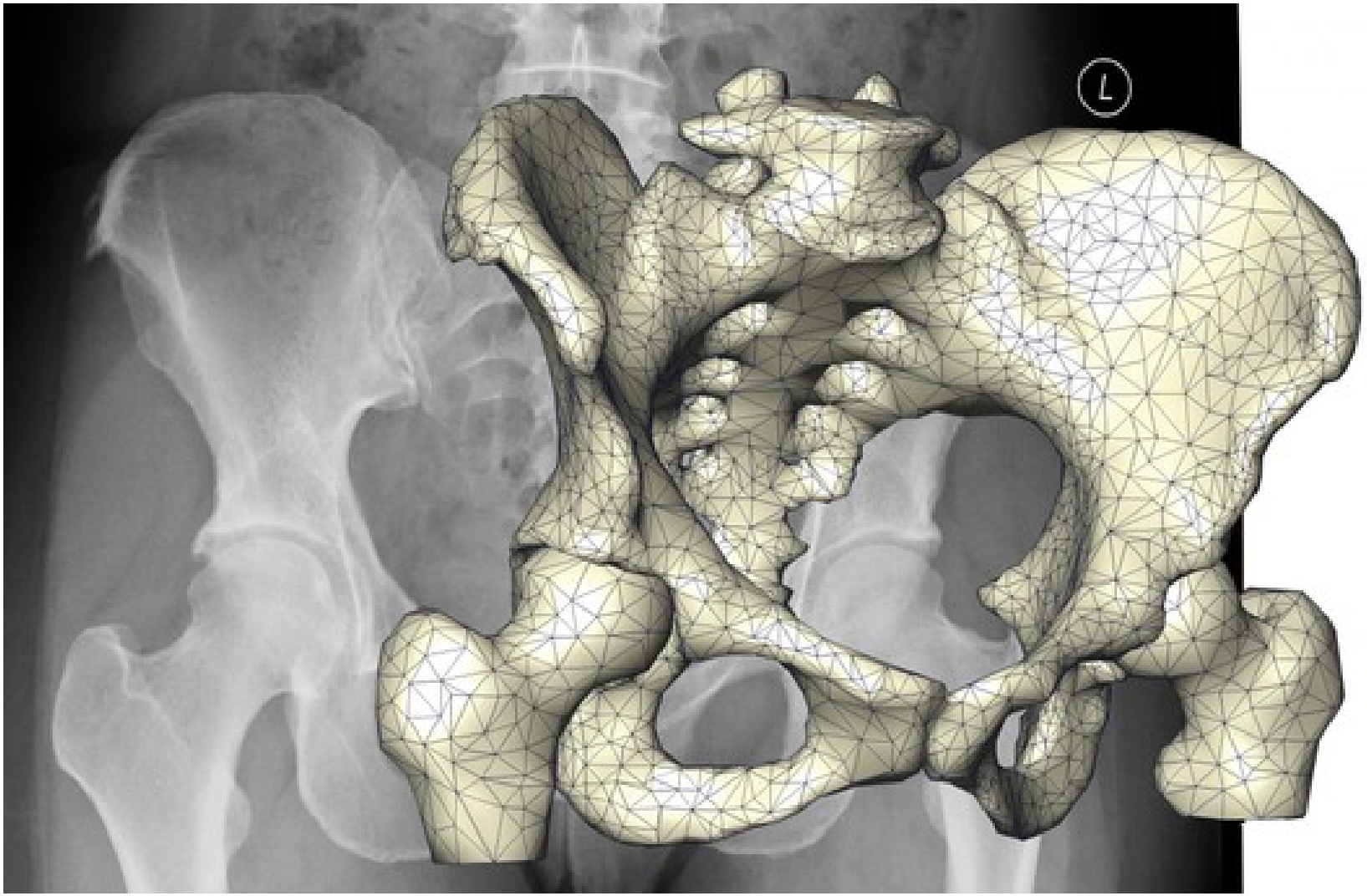
- 3D assessment of tissue and organs from non-invasive scanning
- automated medical image analysis, interpretation, and diagnosis



- neural/computer interface; interpretive prostheses for the blind
- optical character recognition (OCR): recognition of handwritten or printed characters, words, or numbers; e.g. car registration plates

(some computer vision applications and goals, con't)

- 3D reconstruction from radiological scans, and design of prostheses

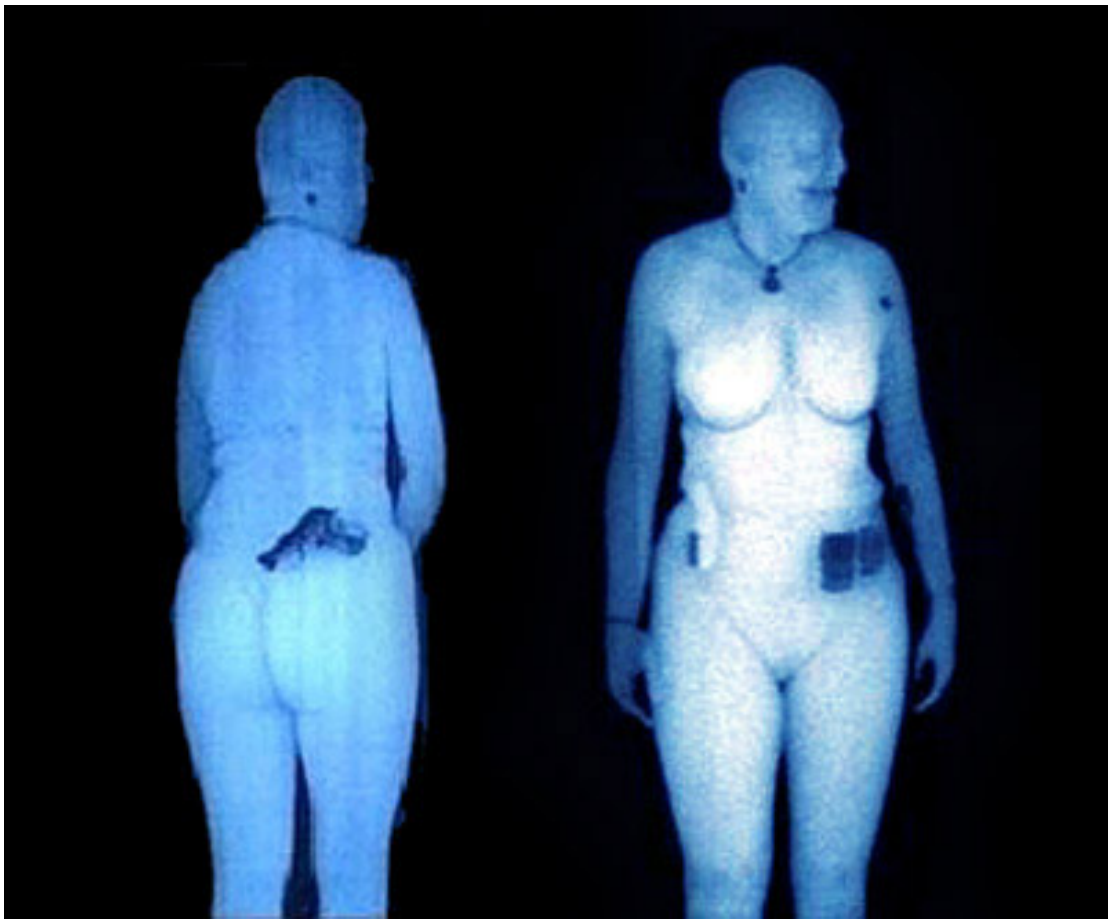


(some computer vision applications and goals, con't)

- robotic manufacturing: manipulation and assembly of parts
- agricultural robots: weeding, harvesting, and grading of produce

(some computer vision applications and goals, con't)

- anomaly detection; event detection; automated surveillance and security screening of passengers at airports



Why the goals of computer vision are so difficult

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, with machine learning;
- and interfacing the representations 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 a 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 understand and to interact intelligently with the world.

(Why the goals of computer vision are so difficult, con't)

Although vision seems like such an effortless, immediate faculty for humans and other animals, it has proven to be 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 must invert the 3D $\rightarrow$ 2D projection in order to recover world properties (object properties in space); but the 3D $\leftarrow$ 2D inversion of such a projection is, strictly speaking, mathematically impossible: there is no unique solution.

   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, gradients, perspective, etc. Vision has to perform exactly the inverse of this process!

   A classic example in computer vision is face recognition. Humans perform this task effortlessly, rapidly, reliably, and unconsciously.

(Why the goals of computer vision are so difficult, con't)

(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)*:

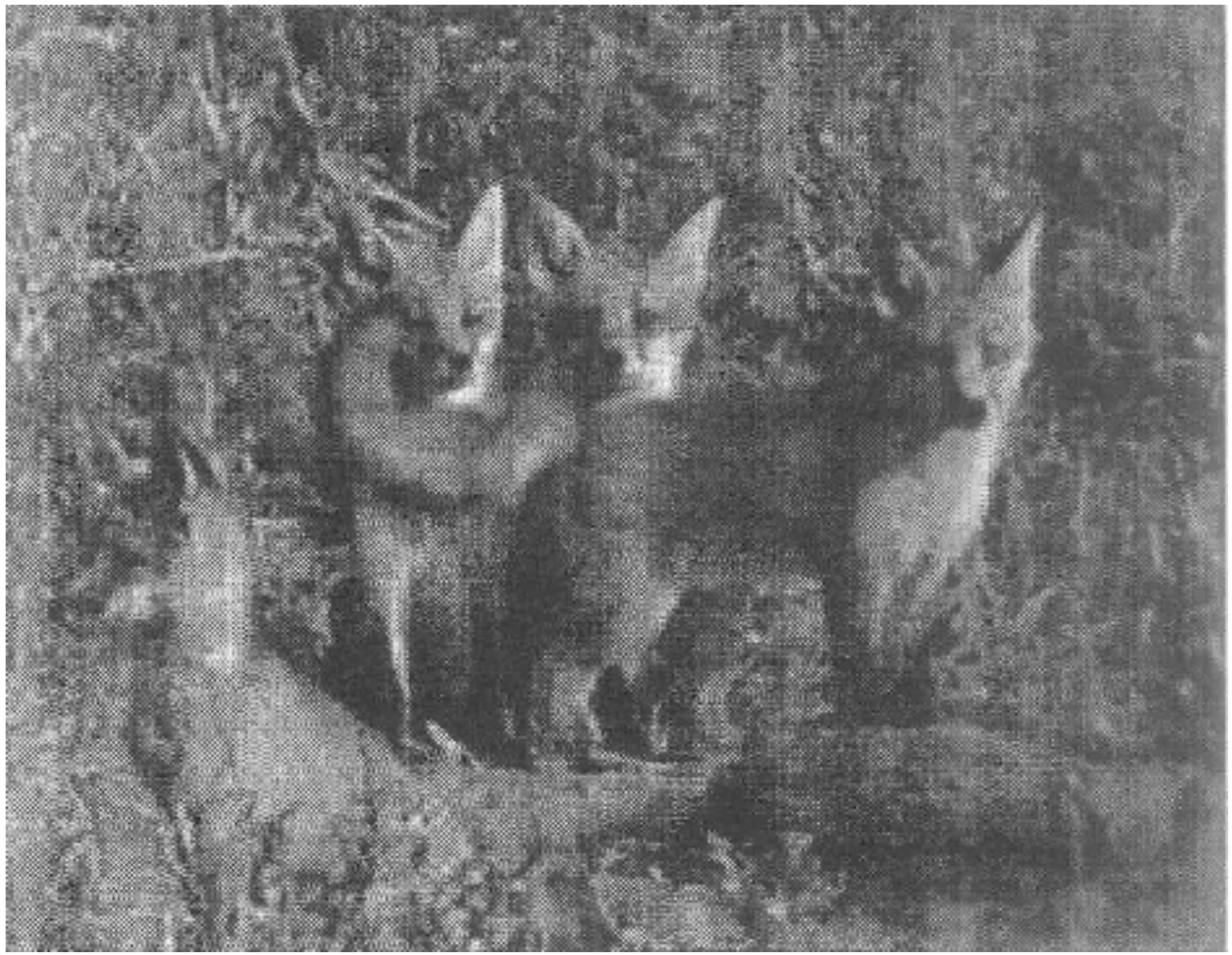
Which two pictures show the same person?

Unlike humans, classical computer vision algorithms would select 1 and 2 as the same person, since those images are more similar than 1 and 3.

However, recently remarkable progress has been made towards achieving good pose-invariant face recognition with Google's "FaceNet", based on a convolutional neural network and "deep learning" from a huge database of hundreds of millions of labelled example face images, in different poses.

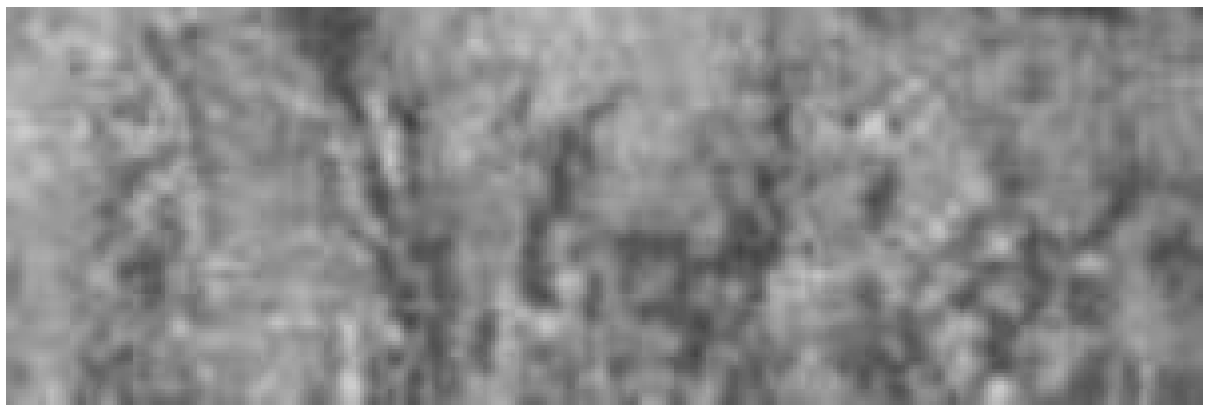
(Why the goals of computer vision are so difficult, con't)

2. Few visual tasks can be performed in a purely data-driven way ("bottom-up" image analysis). Consider this image: the foxes are well camouflaged by their textured backgrounds; the foxes occlude each other; they appear in different poses, perspective angles; etc.



(Why the goals of computer vision are so difficult, con't)

Extracting and magnifying the lower-left corner of the previous image (capturing most of the body of the fourth fox, minus its head) illustrates the impoverished limits of a purely "data-driven, bottom-up" approach.



- How can edge detection algorithms find and trace this fox's outline? Simple methods would meander, finding nonsense edges everywhere.
- Even for humans this is difficult. "Top-down" guidance based on the entire image is needed, allowing the use of prior knowledge about the nature of the world and of the things that may populate it.
- Model-driven vision can drive image parsing by setting expectations. Maybe the three central foxes with their distinctive heads are critical.

(Why the goals of computer vision are so difficult, con't)

The image of foxes was intentionally noisy, grainy, and monochromatic, in order to highlight how remarkable is the fact that we (humans) can easily process and understand the image despite such impoverished data.

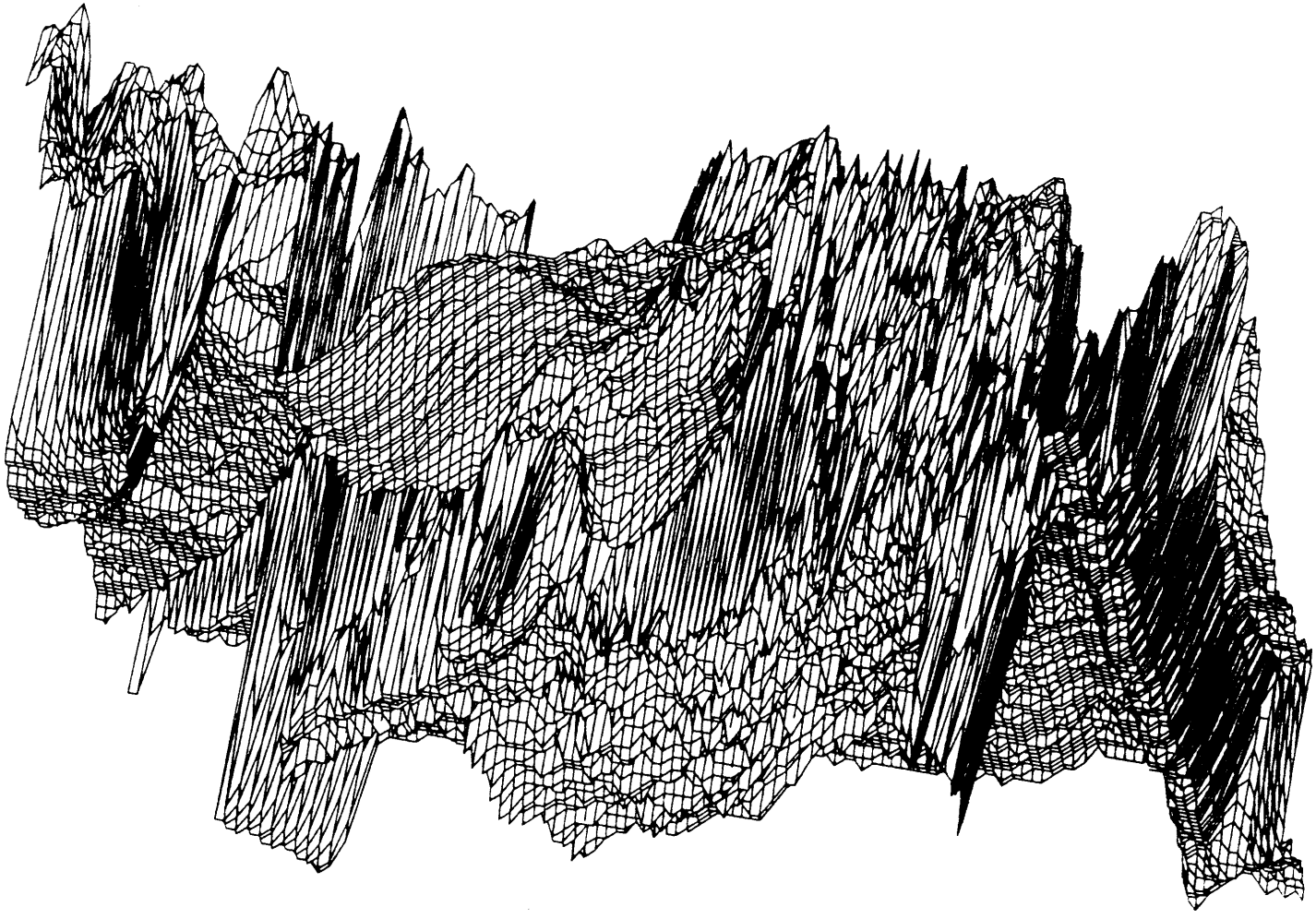
How can there possibly exist mathematical operators for such an image that can, despite its poor quality:

- perform the figure-ground segmentation of the scene (into its objects, versus background clutter)
- 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 2D (single) images, and navigate within them:
http://www.youtube.com/watch?v=VuoljANz4EA .

(Why the goals of computer vision are so difficult, con't)

Consider now the actual image data of a face, shown as a pixel array with greyscale value 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 recognise the face? In this format, the image reveals both the complexity of the problem and the poverty of the signal data.



(Why the goals of computer vision are so difficult, con't)

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

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

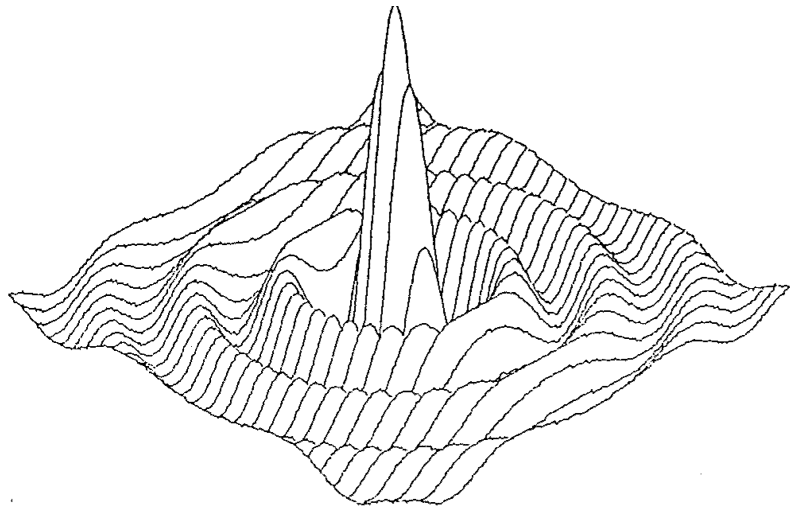
- its solution exists;
- its solution is unique;
- its solution depends continuously on the data.

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

- 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, ...

(Why the goals of computer vision are so difficult, con't)

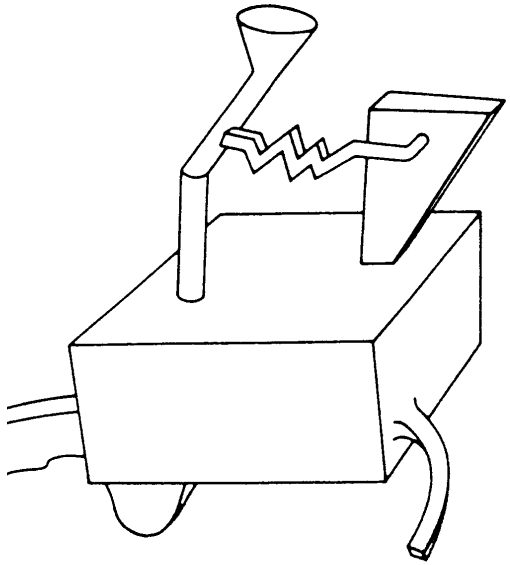
- inferring a 3D shape unambiguously from a 2D line drawing:



- 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): pose-invariant recognition

(Why the goals of computer vision are so difficult, con't)

- understanding an object that has never been seen before:



For a chess-playing robot, the task of visually identifying an actual chess piece in 3D (*e.g.* a knight, with pose-invariance and "design-invariance") is a much harder problem than *playing* chess! (The latter problem was solved years ago, and chess-playing algorithms today perform at almost superhuman skill levels; but the former problem remains barely solved.)

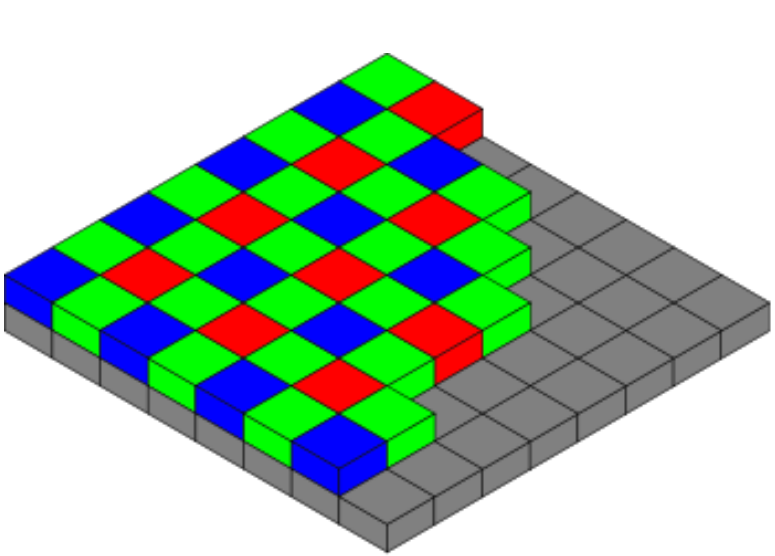
...but enough counsel of despair. Let us begin with understanding what an image array is.

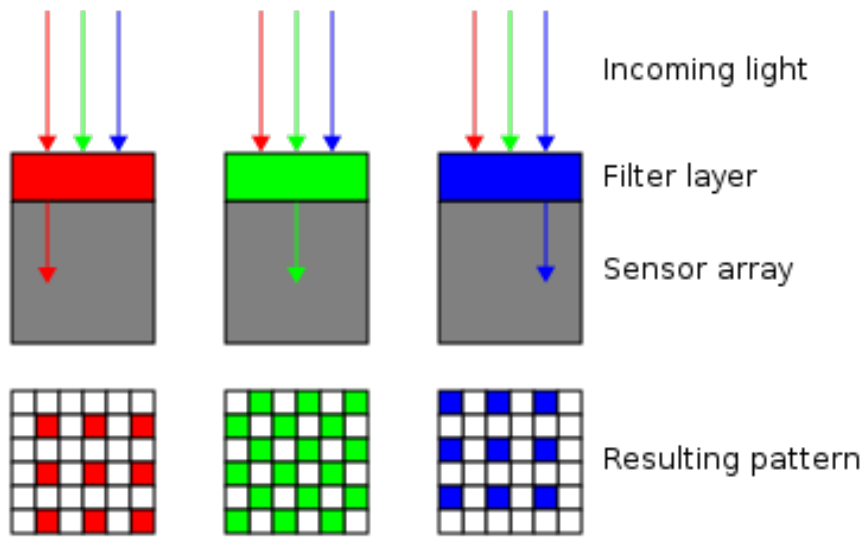
2. Pixel arrays, CCD / CMOS sensors, image coding

- CCD / CMOS cameras contains dense arrays 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. CMOS sensors are simpler, cheaper, and consume only about 1% as much power as CCD sensors, which are more sensitive.
- The number of pixels (picture elements) ranges from a few 100,000 to many millions (*e.g.* 6 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 six times larger than the wavelength of a photon of light in middle of the visible spectrum (yellow $\sim$ 500 nanometers or nm), so quantum mechanics already limits the further resolution possible in sensors sized about 1 cm^2.

(Image sensing, pixel arrays, CCD cameras, con't)

- 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: optical figure-of-merit.
- Luminance resolution (the number of distinguishable grey levels) is determined by the number of bits per pixel resolved by the digitizer, and by the inherent signal-to-noise ratio of the CCD array.
- Colour information arises (conceptually if not literally) from three separate CCD arrays preceded by different colour filters, or mutually embedded as Bayer subpopulations within a single CCD array:





Data in video streams

Composite video uses a high-frequency "chrominance burst" to encode colour; or in S-video there are separate "luma" and "chroma" signals; or there may be separate RGB colour channels. Colour information requires much less resolution than luminance; 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, making a succession of frames.

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, even without HDTV:

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 it encodes?

Image formats and sampling theory

Images are represented as rectangular arrays of numbers (1 byte each), sampling the image intensity at each pixel position. A colour image may be represented in three separate such byte arrays called "colour planes", containing red, green, and blue components as monochromatic images. An image with an oblique edge within it might include this array:

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 for properties of browsers or devices.

Examples of image formats and encodings

- `.jpeg` - for compression of continuous-tone and colour images, with a controllable "quality factor". Tiles of Discrete Cosine Transform (DCT) coefficients are quantised, with frequency-dependent depth.
- `.jpeg2000` - a superior version of `.jpeg` implemented with smooth Daubechies wavelets to avoid block quantisation artifacts.
- `.mpeg` - a stream-oriented, compressive encoding scheme used for video and multimedia. Individual image frames are `.jpeg` compressed, but an equal amount of temporal redundancy is removed by inter-frame predictive coding and interpolation.
- `.gif` - for sparse binarised images; 8-bit colour. Very compressive; favoured for websites and other bandwidth-limited media.
- `.png` - using lossless compression, the portable network graphic format supports 24-bit RGB.
- `.tiff` - A complex umbrella class of tagged image file formats. Non-compressive; up to 24-bit colour; randomly embedded tags.
- `.bmp` - a non-compressive bit-mapped format in which individual pixel values can easily be extracted. Non-compressive.

(Image formats and sampling theory, con't)

- Various colour coordinates are used for "colour separation", such as HSI (Hue, Saturation, Intensity), or RGB, or CMY vector spaces.
- Regardless of the sensor properties and coding format, ultimately the image data must be represented pixel by pixel. For compressed formats, the image payload is actually in a (Fourier-like) transform domain, and so to retrieve an array of numbers representing image pixel values, essentially an inverse transform must be performed on the compressive transform coefficients.
- Typically a monochromatic image is resolved to 8 bits/pixel. This allows 256 different intensity values for each pixel, from black (0) to white (255), with 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 combined payload is only 8 or 12 bits/pixel.

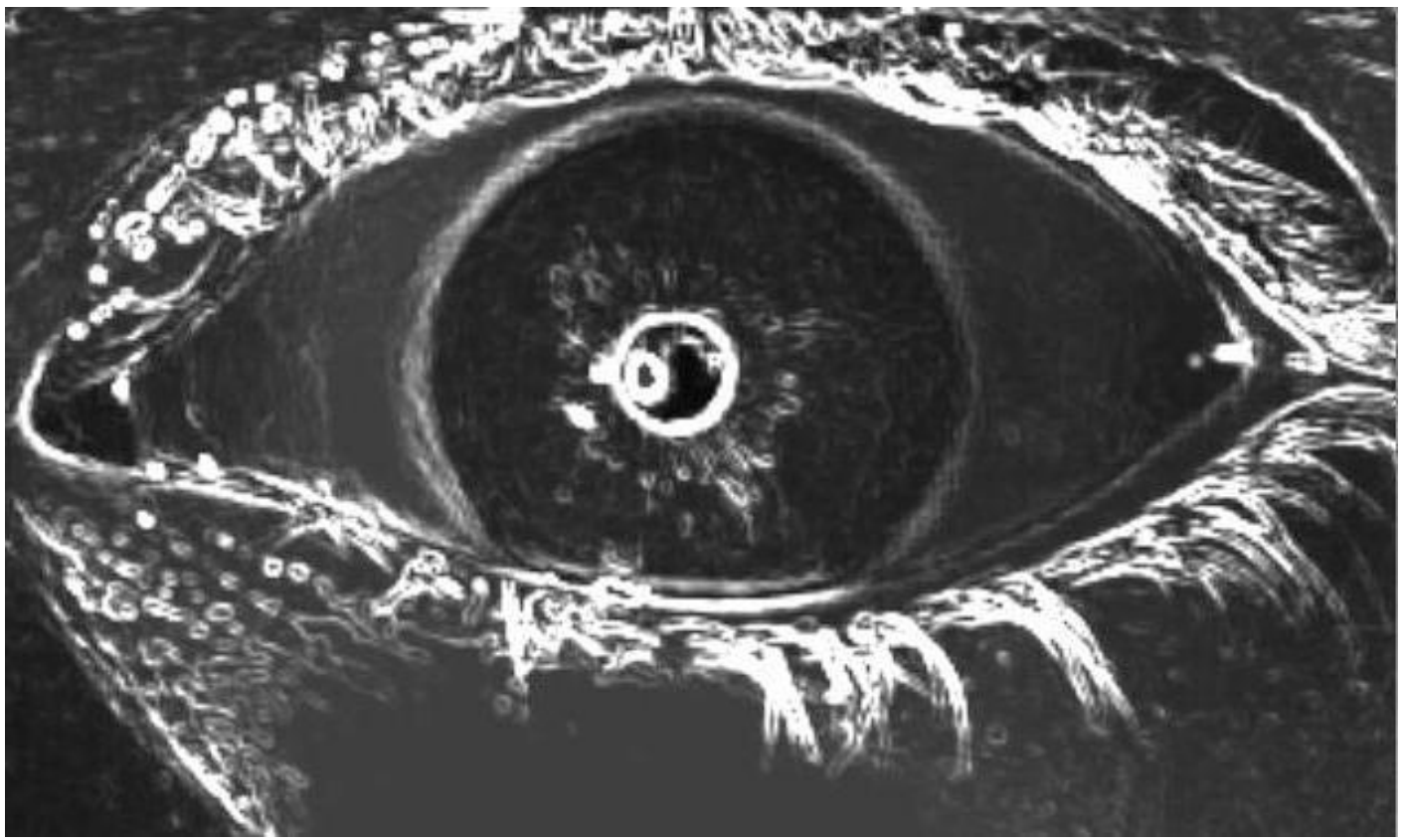
(Image formats and sampling theory, con't)

- How much information does an image contain? Bit count does not relate to optical properties, nor to frequency analysis.
- Nyquist's Sampling Theorem says that the highest *spatial frequency* component of information contained in an image equals one-half the sampling density of the pixel array.
- Thus a pixel array with 640 columns can represent spatial frequency components of image structure no higher than 320 cycles/image.
- Likewise, if image frames are sampled in time at 30 per second, then the highest *temporal frequency* component of information contained within a moving sequence is 15 Hertz.
- Increasingly, more complex sensors called RGB-D sensors capture colour as well as depth information for purposes such as human activity recognition, tracking, segmentation, and 3D reconstruction from RGB-D data.

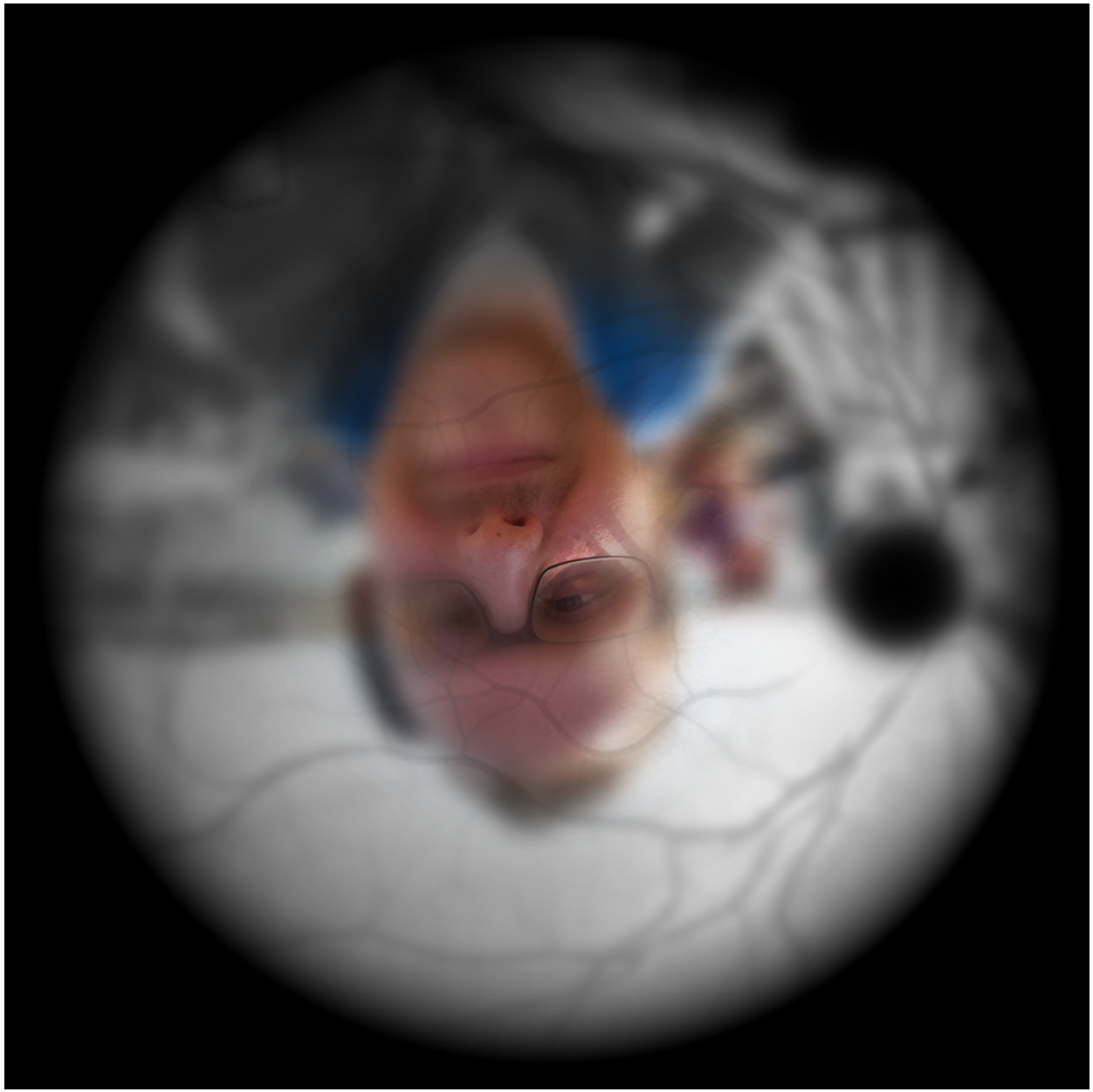
  State-of-the-art (2018) reference on RGB-D (3D) reconstruction:
  https://web.stanford.edu/~zollhoef/papers/EG18_RecoSTAR/paper.pdf

Using second-order pixel statistics to assist segmentation

- Low-level local statistical metrics can be useful for segmentation (dividing an image into meaningful regions). For example, in the NIR band (700nm – 900nm) used for iris imaging, it can be difficult to detect the boundary between the eye's sclera and the eyelid skin.
- But after computing pixel variance and mean in local (4×4) patches, imaging their *ratio* sets the eyelid boundaries (eyelashes) "on FIRE". This helps with eye-finding because of the distinctive, iconic, orifice.

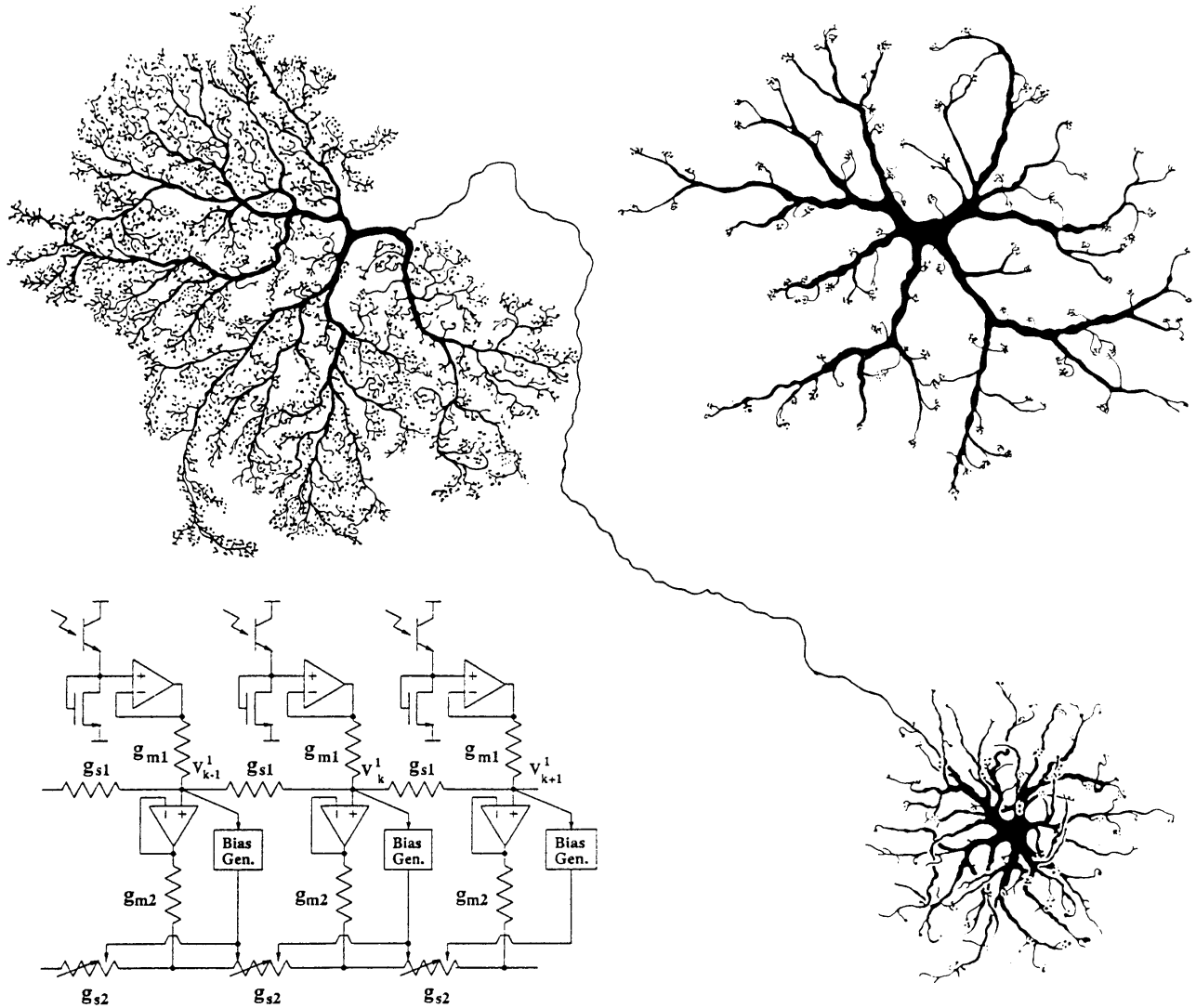


3. Biological visual mechanisms: retina to visual cortex



Low-level neurobiological mechanisms

- No artificial 'general purpose' vision system has yet been built.
- Natural vision systems abound. What can we learn from visual neuroscience, despite the enormous differences in hardware?



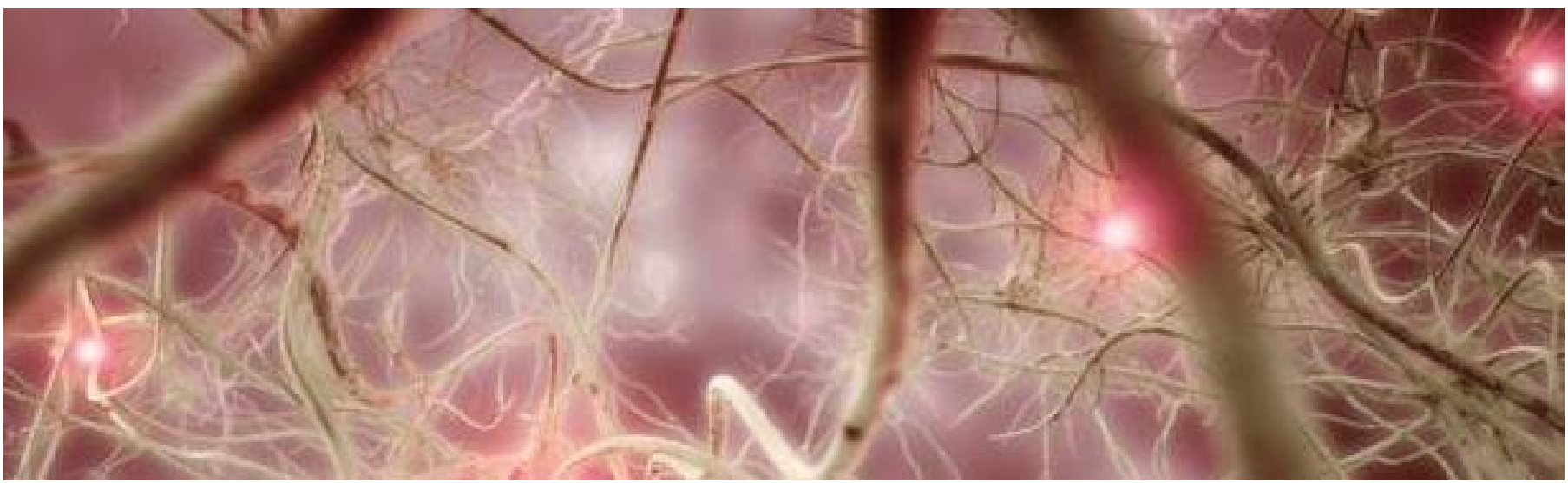
Wetware

- Neurones are sluggish but richly interconnected cells having both analogue and discrete aspects, with nonlinear, adaptive features.
- Fundamentally they consist of an enclosing membrane that can separate electrical charge, so a voltage difference generally exists between the inside and outside of a neurone.
- The membrane is a lipid bilayer that has a capacitance of about 10,000 μFarads/cm^2, and it also has pores that are differentially selective to different ions (mainly Na^+, K^+, and Cl^-). Seawater similarity: – *L'homme vient de l'océan, son sang reste salé.*
- These ion species cross the neural membrane through protein pores acting as discrete conductances (hence as resistors).
- The resistors for Na^+ and K^+ have the further crucial property that their resistance is not constant, but voltage-dependent.
- As more positive ions (Na^+) flow into the neurone, the voltage becomes more positive on the inside, and this further reduces the membrane's resistance to Na^+, allowing still more to enter.
- This catastrophic breakdown in resistance to Na^+ constitutes a nerve impulse. Within about a msec a slower but opposite effect involving K^+ restores the original trans-membrane voltage.

(Wetware, con't)

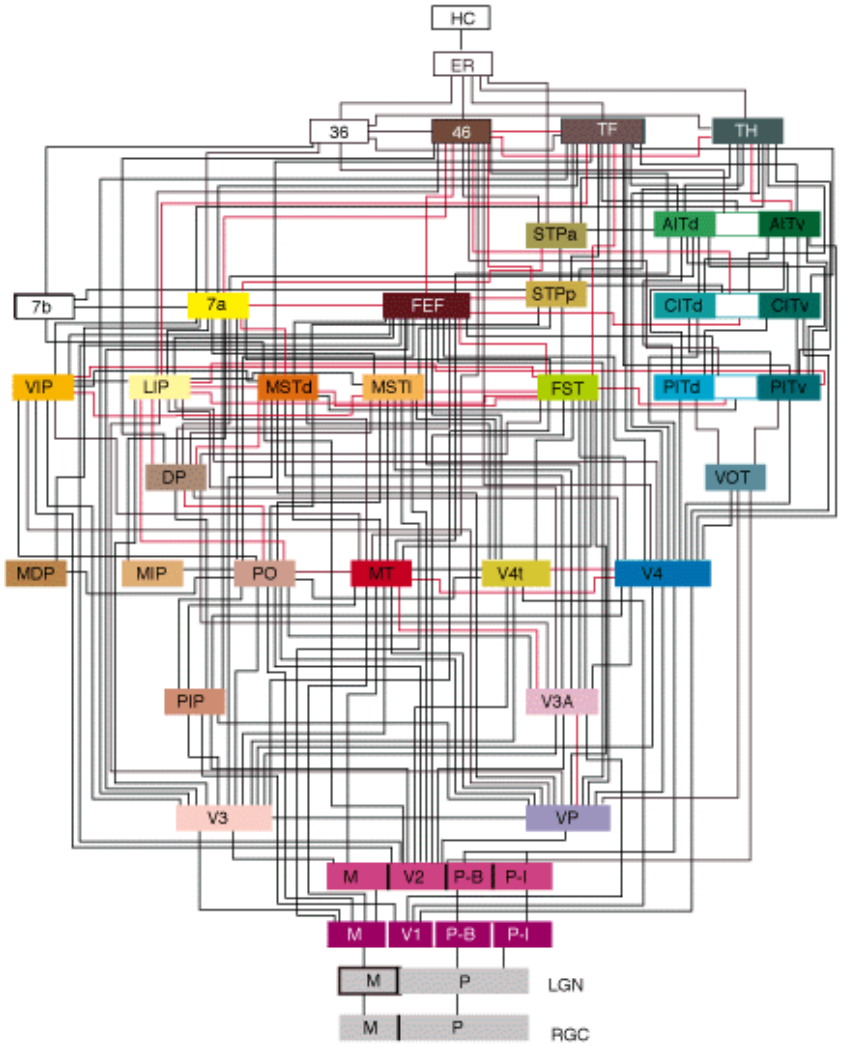
- After a refractory period of about 2 msec to restore electro-osmotic equilibrium, the neurone is ready for action again.
- Nerve impulses propagate down axons at speeds of about 100 m/sec.
- Impulse signalling can be described as discrete, but the antecedent summations of current flows into a neurone from other neurones at synapses, triggering an impulse, are essentially analogue events.
- In general, neural activity is fundamentally asynchronous: there is no master clock on whose edges ⊓⊔⊓⊔ the events occur.
- Impulse generation time prevents "clocking" faster than $\sim$ 300 Hz (– about 10 million times slower than the 3 GHz clock in your PC.)
- Balanced against this sluggishness is massive inter-connectivity:
- Typically in brain tissue, there are about 10^5 neurones / mm^3.
- Each has $10^3 - 10^4$ synapses with other neurones within $\sim$ 3 cm.
- Thus brain tissue has about 3 kilometers of "wiring" per mm^3 !
- Not possible to distinguish between processing and communications, as we do in Computer Science. They are inseparable.

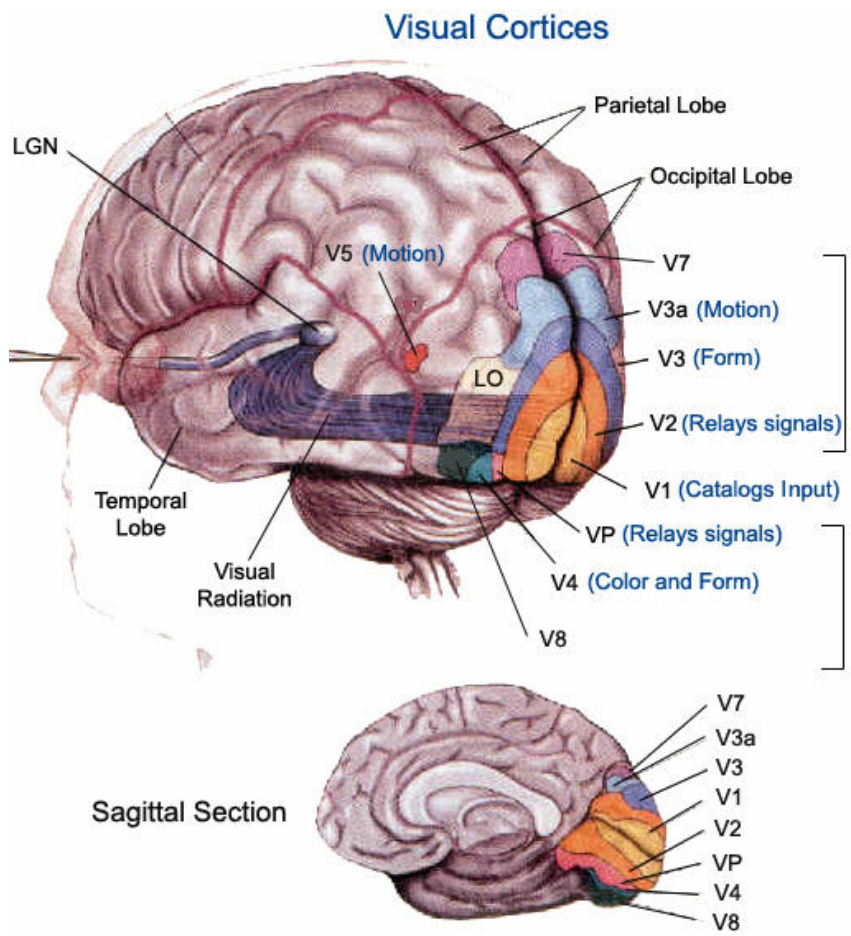
Try to grasp that statistic: 3 kilometers of "wiring" per cubic-millimetre. (Graphical depictions of axonal arborisation and dendritic arborisation):



Mapping 3D brain wiring (video): https://www.youtube.com/watch?v=Bjd8nPKKIU4

- Human brain has about 10^{11} neurones, making about 10^{15} synapses.
- About $2/3^{rds}$ of the brain receives visual input; we are fundamentally visual creatures. There are at least 30 different visual areas, with reciprocal connections, of which the primary visual cortex in the occipital lobe at the rear has been the most extensively studied.

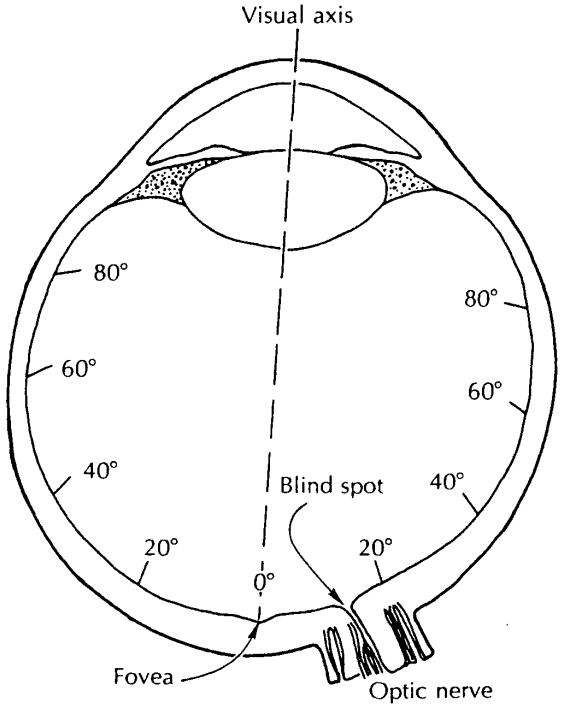


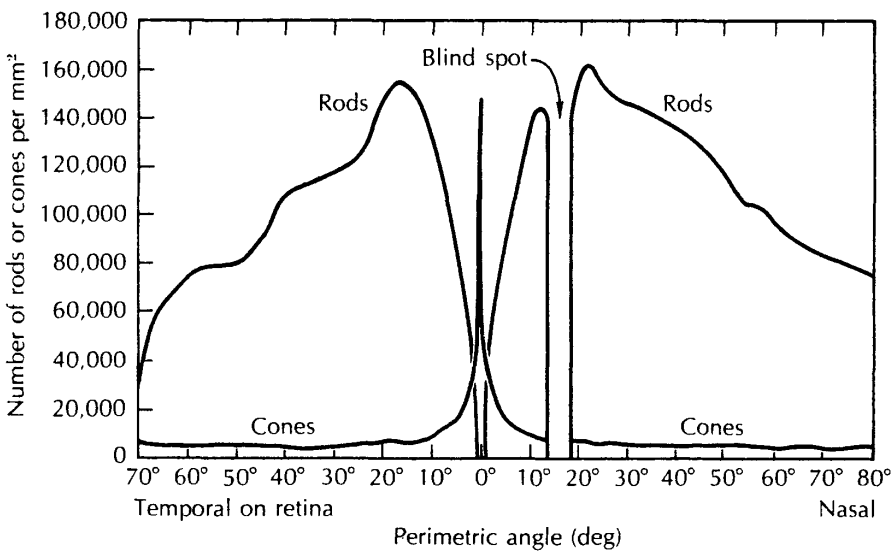


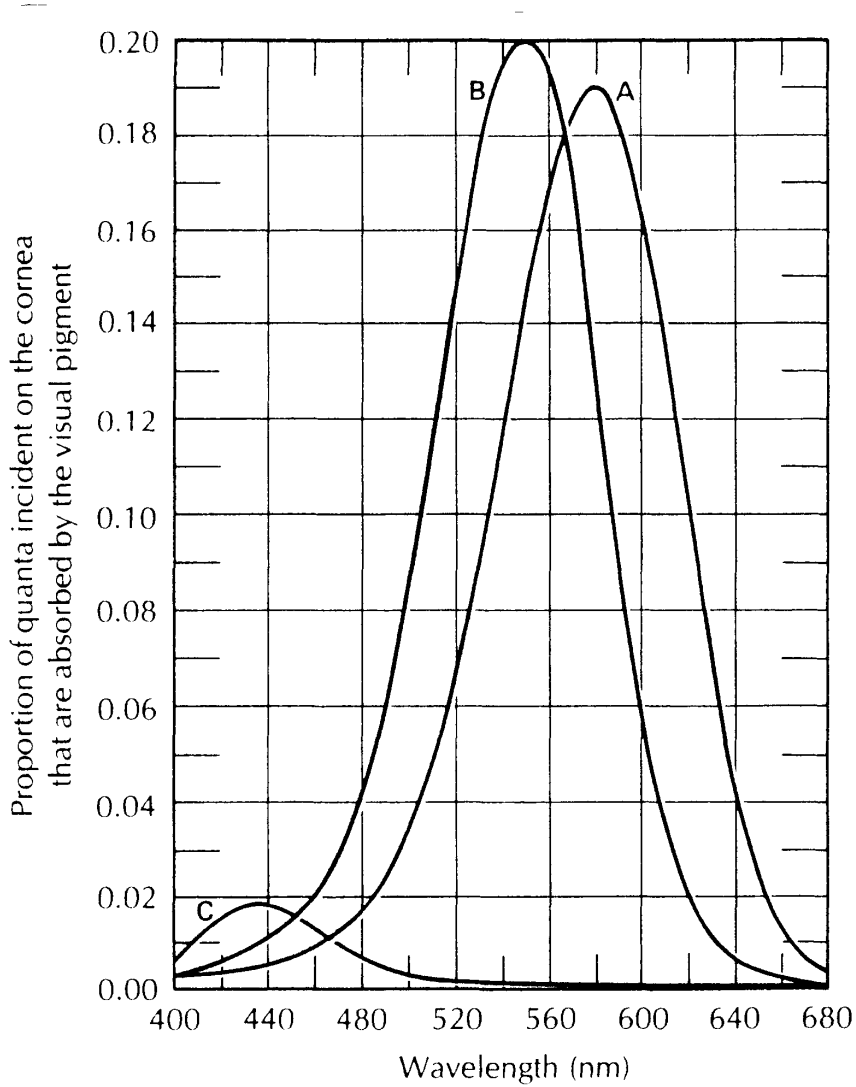
The retina

- The mammalian eye is formed as an extruded vesicle of the brain.
- The retina is about 1 mm thick and it contains about 120 million light-sensitive photoreceptors, of which only 6 million are cones (with photopigments specialised for red, green, or blue wavelengths). The rest are rods which do not discriminate in wavelength bands.
- The visible spectrum of light has wavelength range 400nm - 700nm.
- Rods are specialised for much lower light intensities. They subserve our "night vision" (hence the absence of perceived colour at night), and they pool their responses, at the cost of spatial resolution.
- Cones exist primarily near the fovea, in about the central 20° where their responses are not pooled, giving much higher spatial resolution.
- As cones function only at higher light levels, we really have a dual system with two barely overlapping sensitivity 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 lowest level, 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).

Distributions of photoreceptors

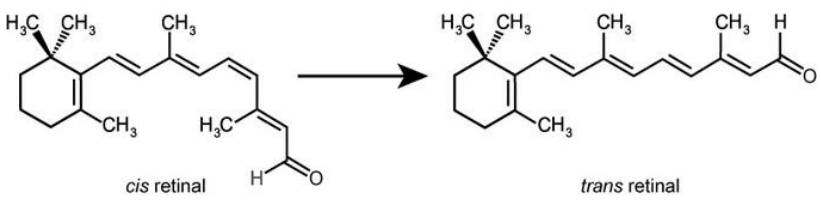




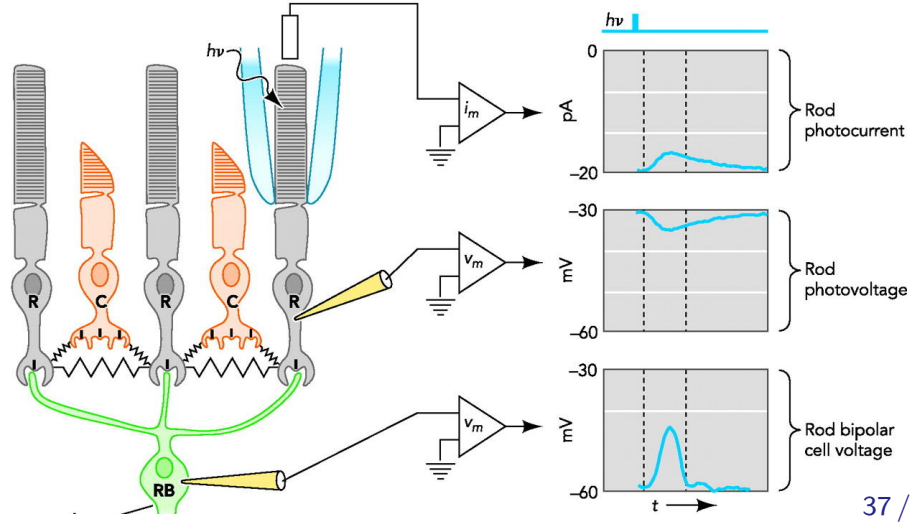


Phototransduction and colour separation

- The most distal neurones in the retina are analogue devices.
- Photoreceptors do not generate impulses but respond to absorption of photons by hyperpolarisation (increased trans-membrane voltage).
- This happens because in the photo-chemical isomerisation reaction, 11-cis-retinal + $\hbar\nu \rightarrow$ all-trans-retinal, a carbon double-bond simply flips from cis to trans, and this causes a pore to close to Na^+ ions.
- As Na^+ ions are actively pumped (the Na^+ "dark current"), this increased resistance causes an increased trans-membrane voltage.
- Voltage change is sensed synaptically by bipolar and horizontal cells.
- The three colour-selective classes of cones have cis-retinal embedded in different opsin molecules. These quantum-mechanically affect the probability of photon capture as a function of wavelength $\lambda = c/\nu$.

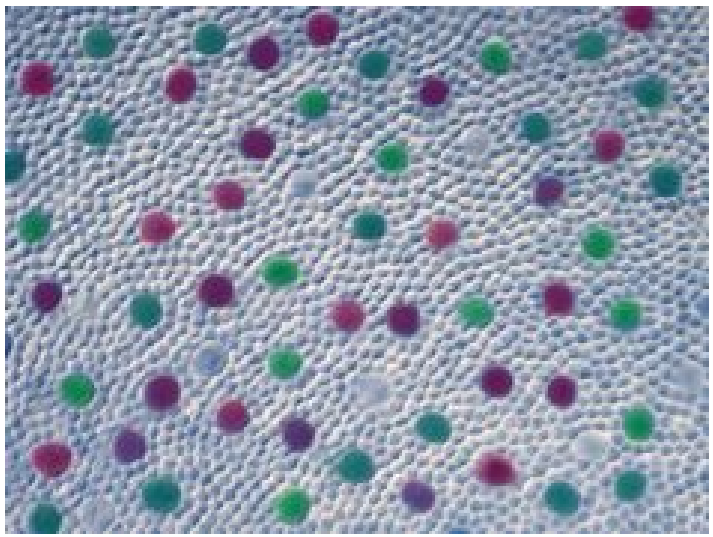
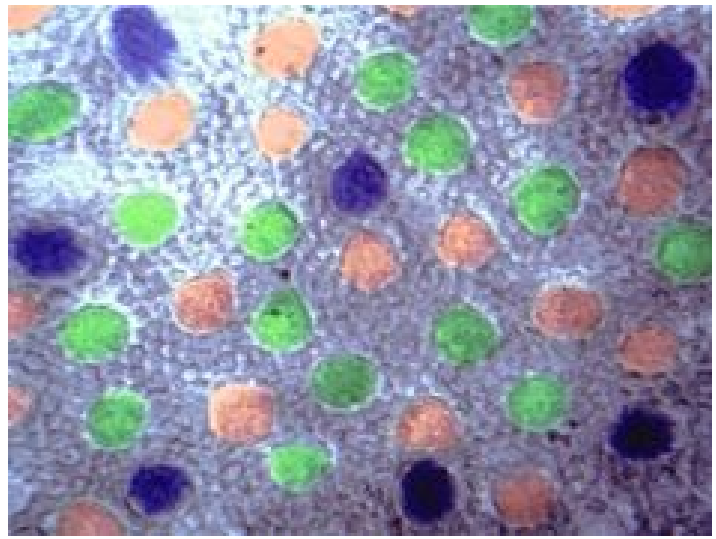
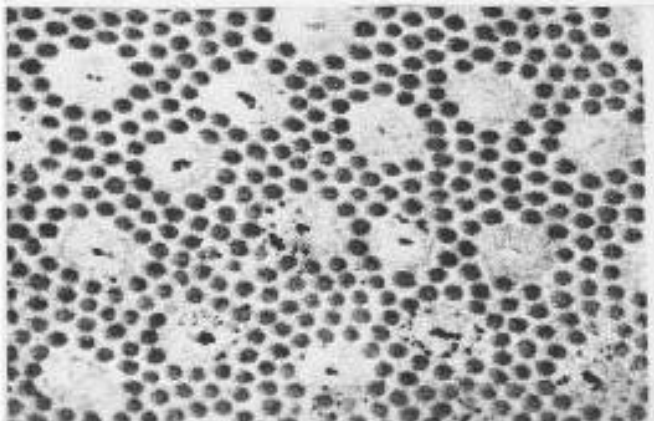
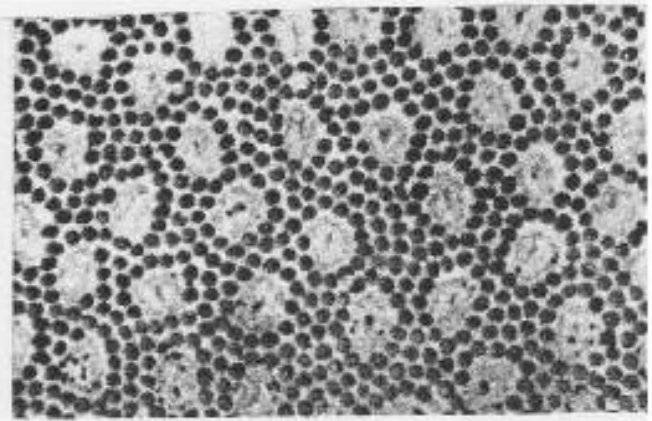


Units: λ in nm (10^{-9}m); ν in Hz (1/sec);
c = speed of light = 3×10^8 m/sec.

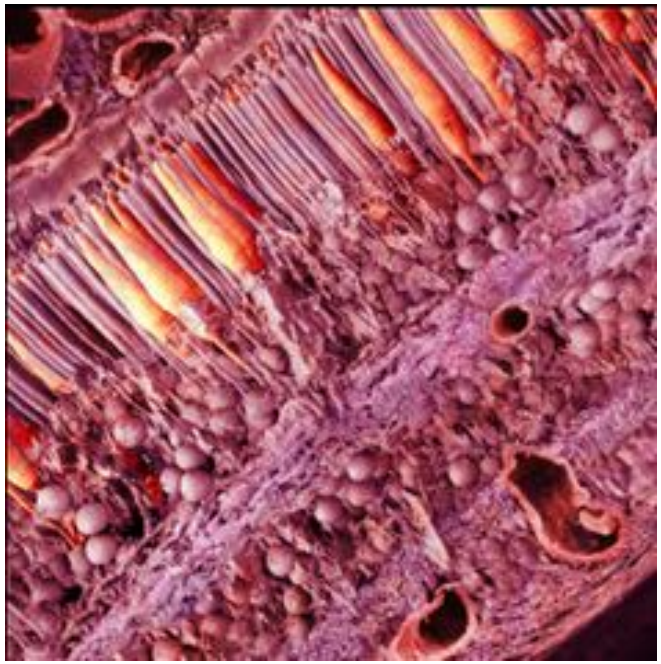


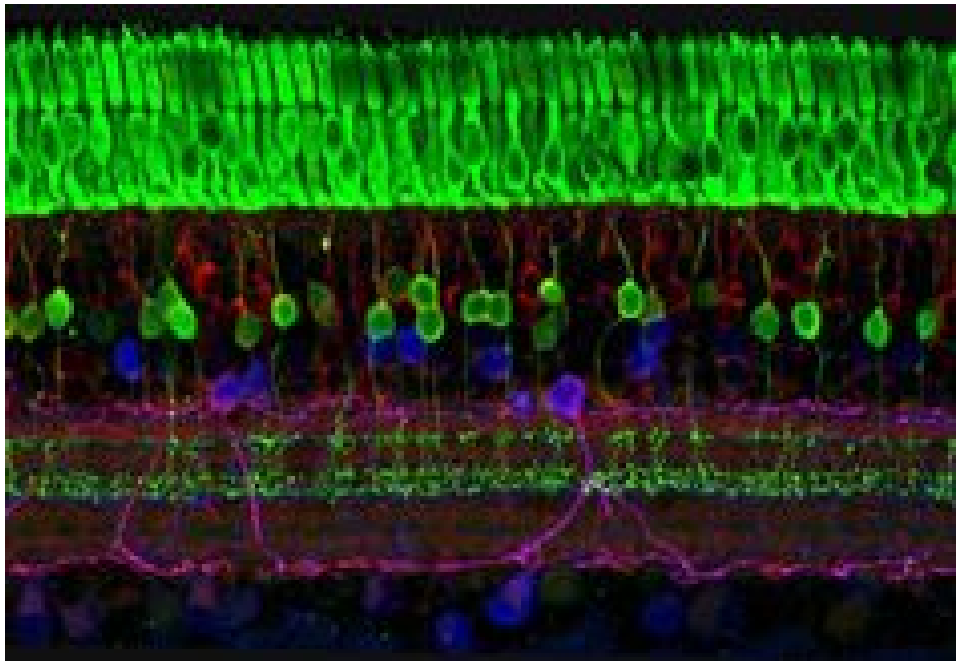
Photoreceptor sampling arrays

- Rods and cones are distributed across the retina in approximately hexagonal lattices but with varying relative densities, depending on eccentricity (distance from the fovea, measured in degrees of arc).
- The hexagonal lattices are imperfect (incoherent, not crystalline), which helps to prevent aliasing of high resolution information.

Retina: not a sensor, but a part of the brain

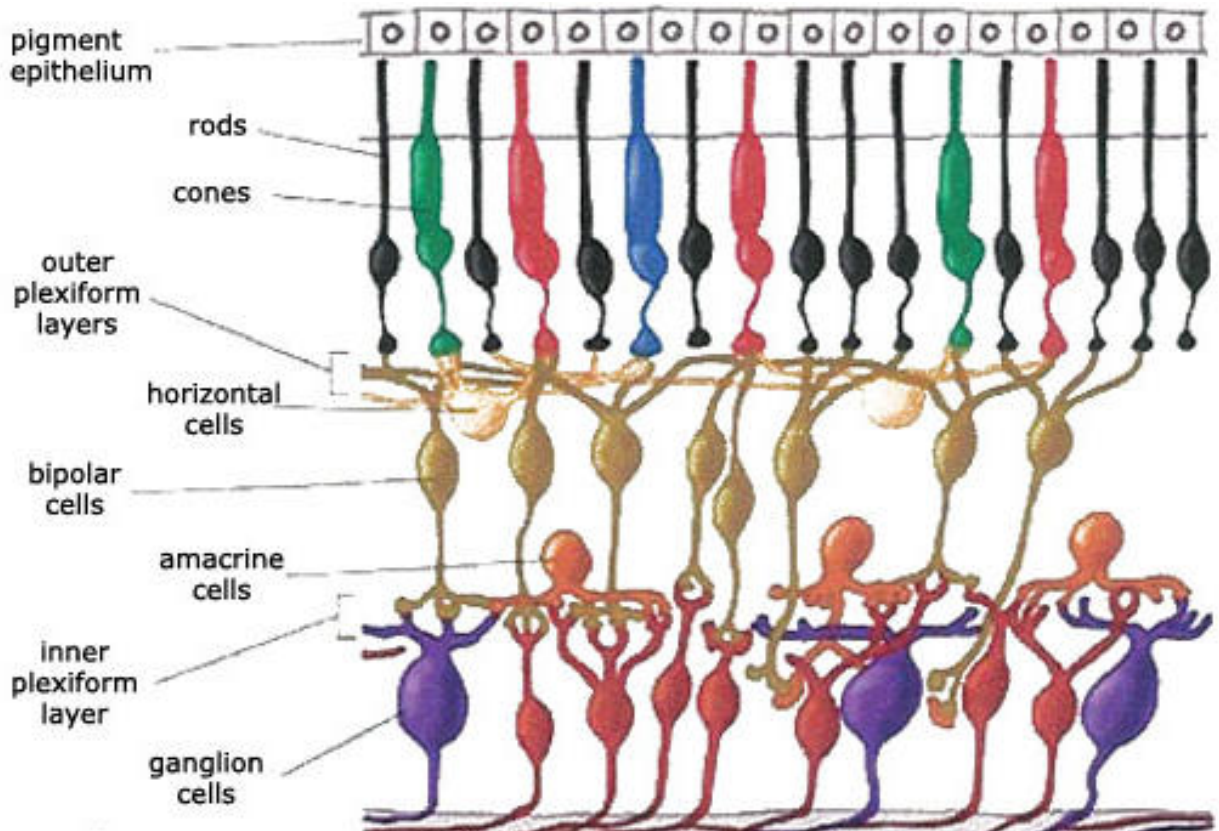




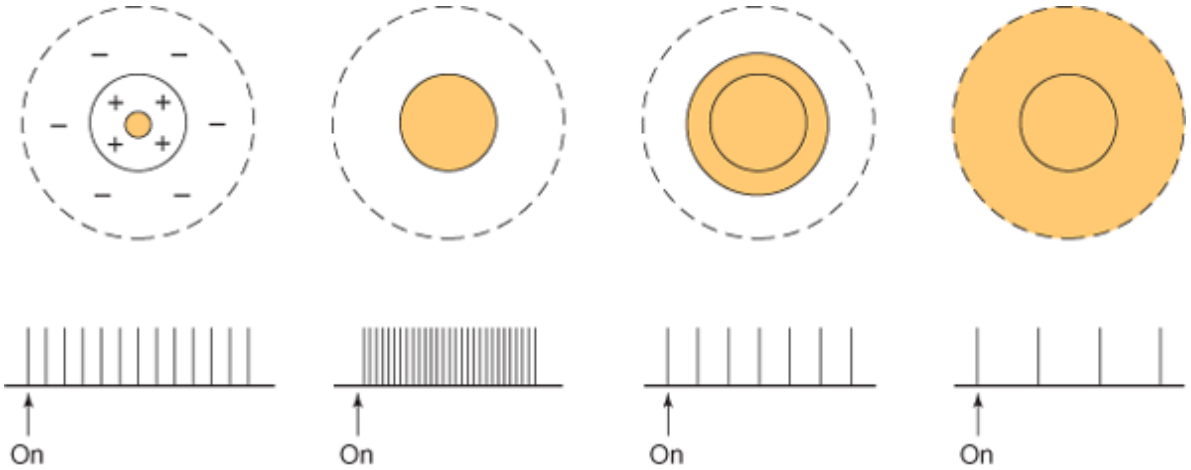
- Far from acting as a camera, the retina is actually part of the brain.
- Note that there are 120 million photoreceptors, but only 1 million "output channels" (the axons of the ganglion cells which constitute the fibres of the optic nerve, sending coded impulses to the brain).
- Actual retinal cross-sections above (with flourescent dyes and stains) reveal some of the complexity of retinal networks.
- Already at its first synapse, the retina is performing a lot of spatial image processing, with temporal processing at the second synapse.

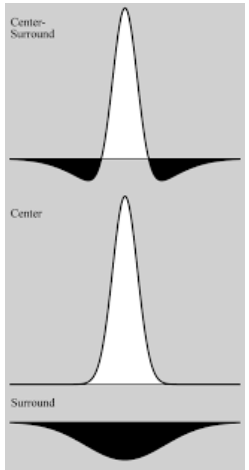
Lateral and longitudinal signal flows in the retina

- The retina is a multi-layered network, containing three nuclear layers (of neurones) and two plexiform layers (synaptic interconnections).
- Paradoxically, the photoreceptors are at the rear, so light must first travel through all of the rest of the retina before being absorbed.
- There are two orthogonal directions of signal flow in the retina: longitudinal (photoreceptors $\rightarrow$ bipolar cells $\rightarrow$ ganglion cells); and lateral (horizontal and amacrine cells, outer/inner plexiform layers).

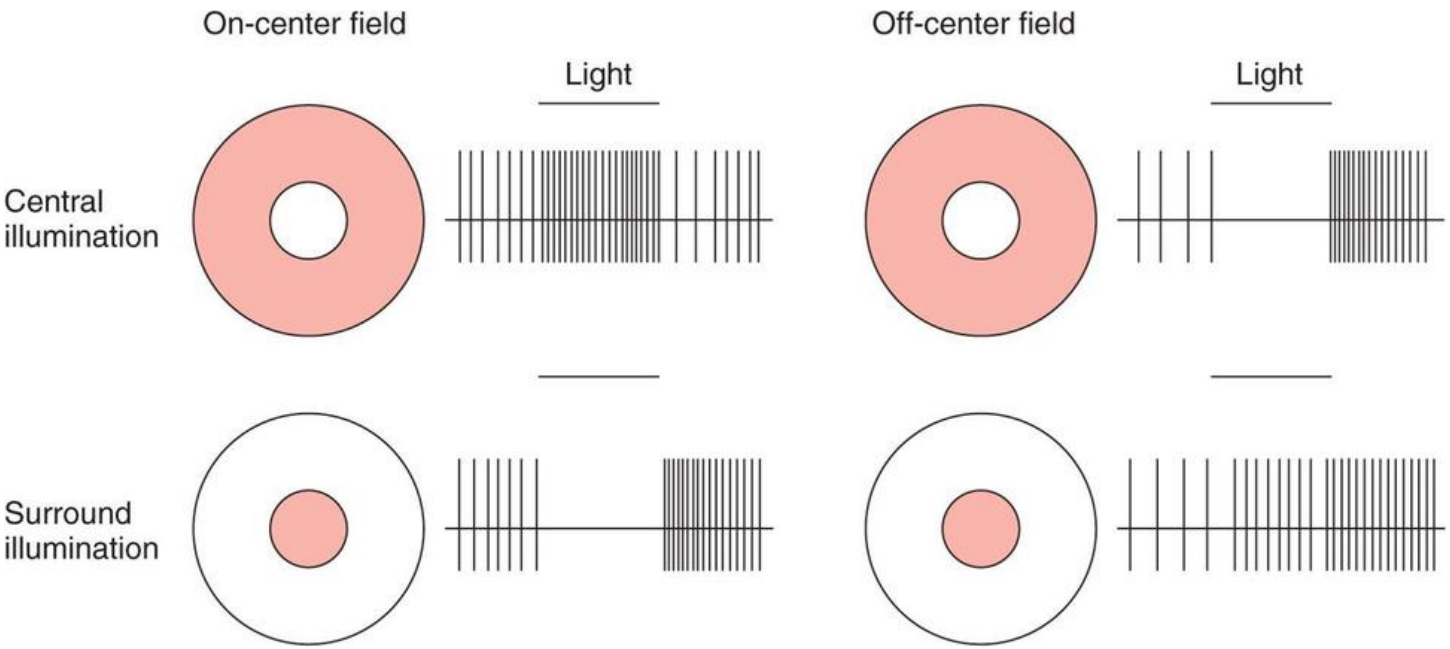


Centre-surround opponent spatial processing in the retina



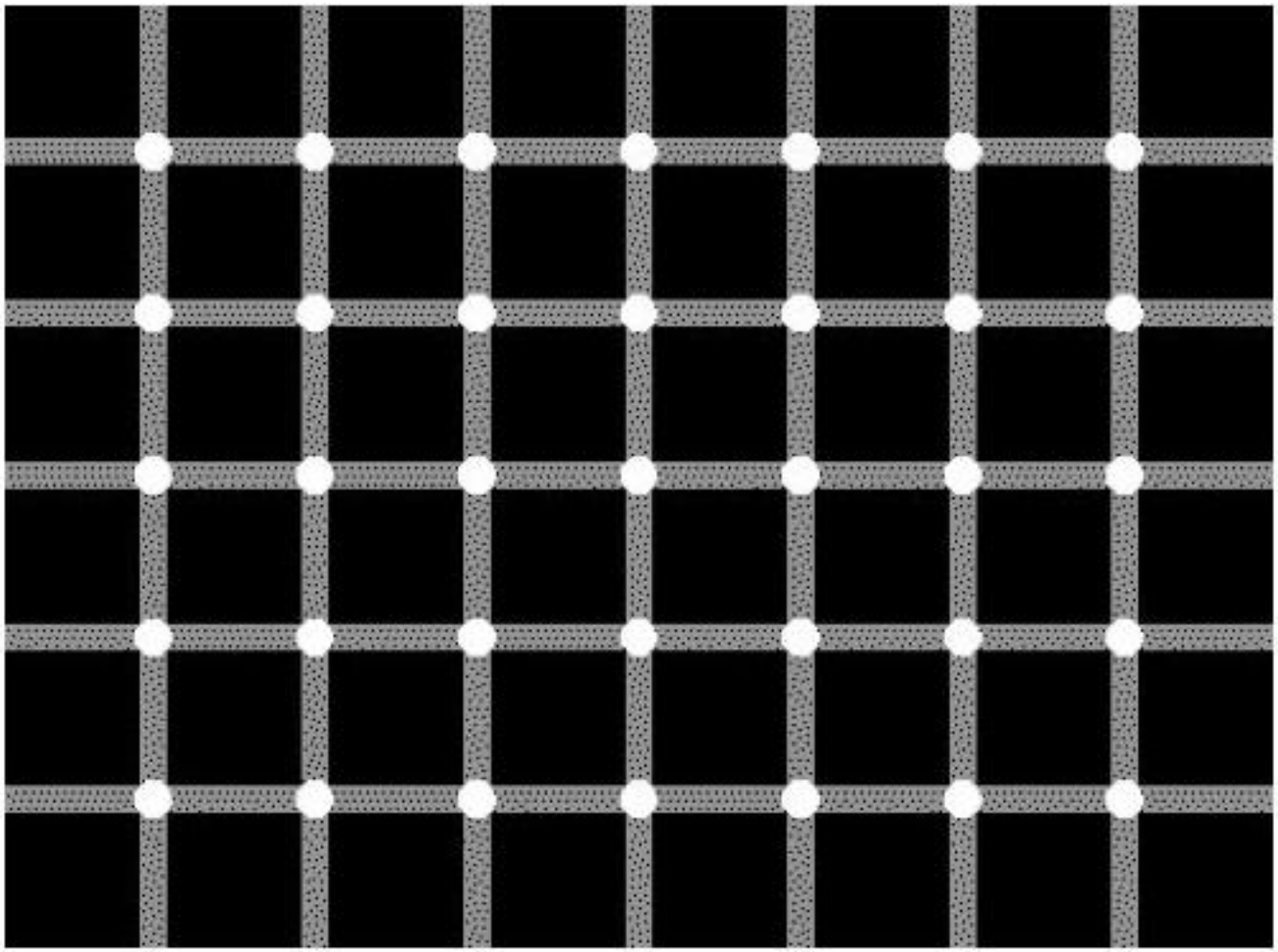


- Excitatory and inhibitory spatial structure creates a bandpass filter
- Such linear filters exist in both polarities: "on-centre" or "off-centre"



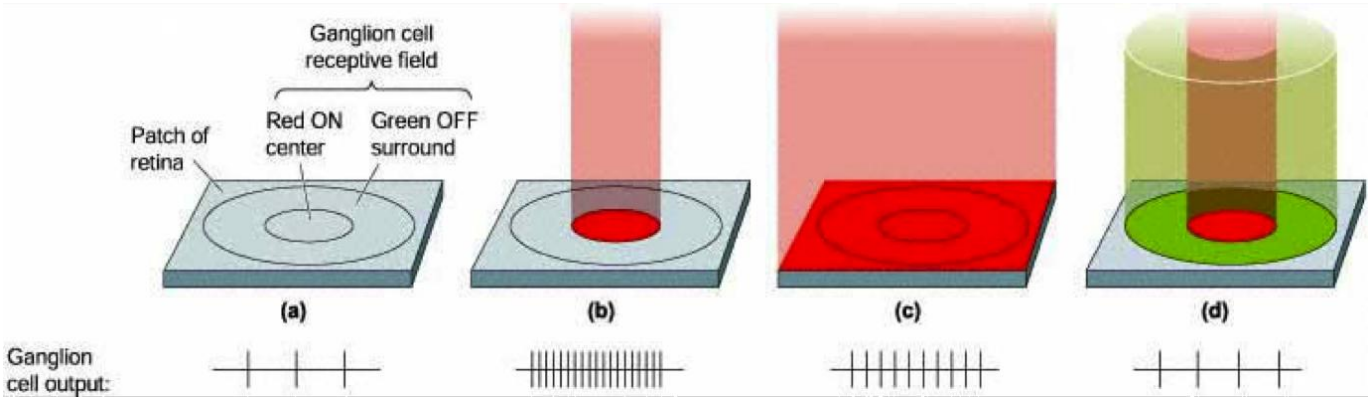
"See" your own retinal centre-surround operators!

- There actually are no dark circles at the intersections in this grid.
- But just move your eyes around the image, and you will see brief round flashes of illusory darkness at those positions, thanks to surround inhibition in the receptive fields of your retinal neurones.



Summary of image processing and coding in the retina

- Sampling by photoreceptor arrays, with pooling of signals from rods
- Both convergence ("fan-in") and divergence ("fan-out") of signals
- Spatial centre-surround comparisons implemented by bipolar cells (direct central input from photoreceptors, minus surround inhibition via horizontal cells, in an annular structure having either polarity)
- Temporal differentiation by amacrine cells, for motion detection
- Separate channels for sustained *versus* transient image information by different classes of ganglion cells (parvo-cellular, magno-cellular)
- Initial colour separation by "opponent processing" mechanisms (yellow *versus* blue; red *versus* green) sometimes also coupled with a spatial centre-surround structure, termed "double opponency"



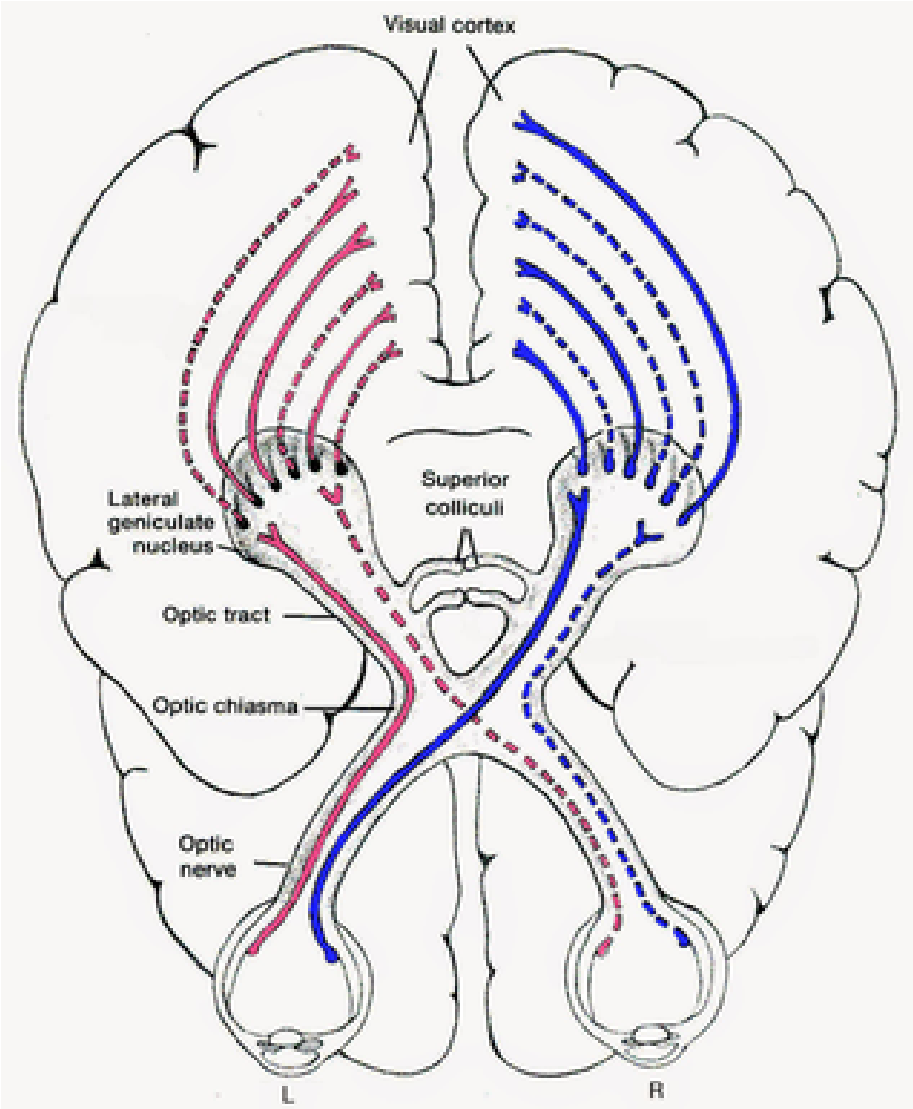
- Generation of nerve impulses in a parallel temporal modulation code, transmitted down the 1 million optic nerve fibres leaving each eye

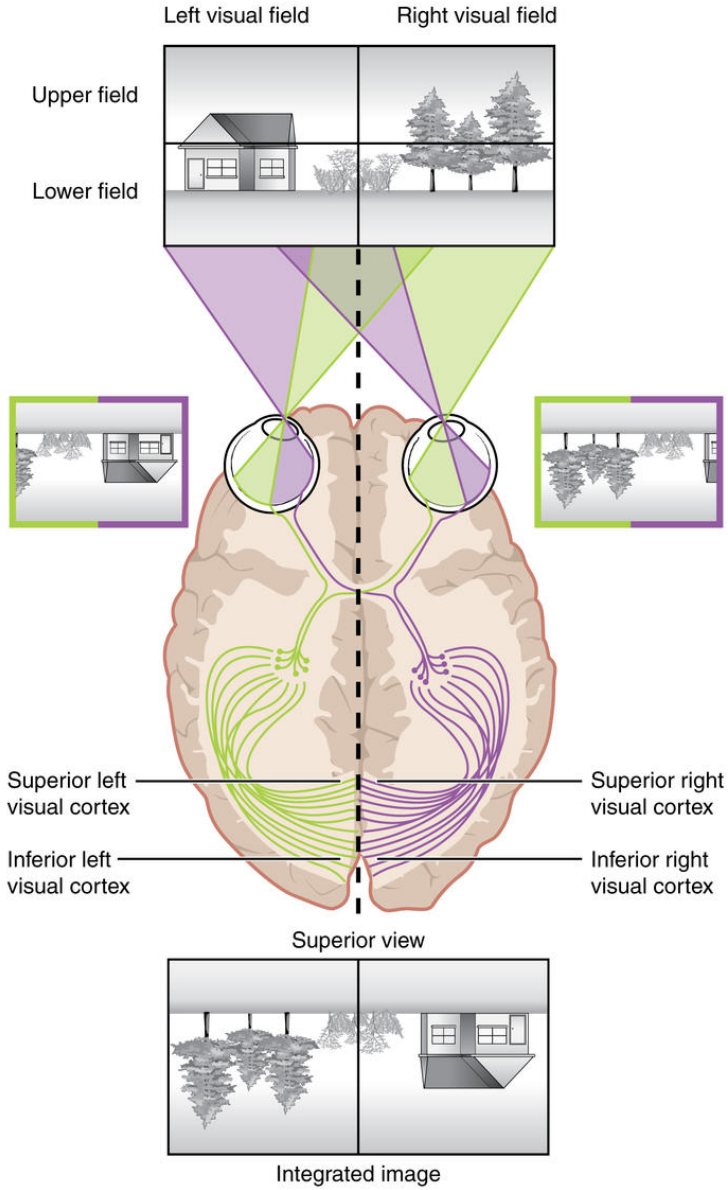
"Receptive field profile" as an image operator

- The visual area to which a neurone responds is its receptive field.
- The spatial profile of excitatory and inhibitory influences on a given visual neurone determines its properties as a spatial image operator.
- Likewise for the temporal structure of neural responses.
- In both space and time, retinal neurones can be described as filters whose response profiles are convolved with the visual input.
- To the extent that they act as linear devices (proportionality and superposition of responses to components of stimuli), their behaviour can be understood (and predicted for arbitrary images) through Fourier analysis and the other tools of linear systems analysis.
- Visual neurones act as integro-differential image operators, with specific scales and time constants, and may be direction-specific.
- An important aspect of retinal receptive fields – as distinct from those found in most neurones of the visual cortex – is that their field structure is quite isotropic (circularly symmetric), not oriented.
- New processing variables are introduced in the brain's visual cortex.

Brain projections and visual cortical architecture

- The right and left visual fields project to different brain hemispheres.





Visual splitting and cortical projections

- You actually have *two quasi-independent* brains, not one.*
- The optic nerve from each eye splits into two at the optic chiasm.
- The portion from the nasal half of each retina crosses over to project only to the contralateral (opposite side) brain hemisphere.
- The optic nerve portion bearing signals from the temporal half of each eye projects only to the ipsilateral (same side) brain hemisphere.
- Therefore the left-half of the visual world (relative to gaze fixation) is directly seen only by the right brain, while the right-half of the visual world is directly seen only by the left brain.
- It is not unreasonable to ask why we don't see some kind of "seam" going down the middle...
- Ultimately the two brain hemispheres share all of their information via a massive connecting bundle of 500 million commissural fibres called the corpus callosum.

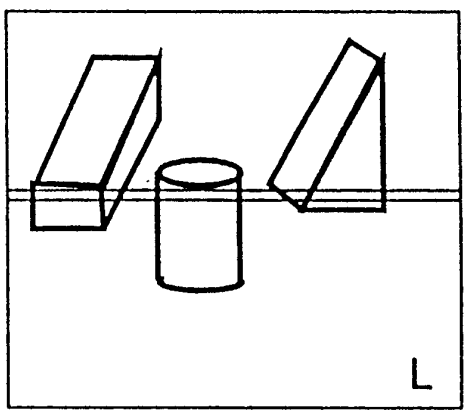
*Commissurotomy, a radical surgical "last resort" for epilepsy patients in the 1960s, separated the hemispheres, allowing two minds to co-exist. [C.E. Marks, *Commissurotomy...*, MIT Press, 1986.]

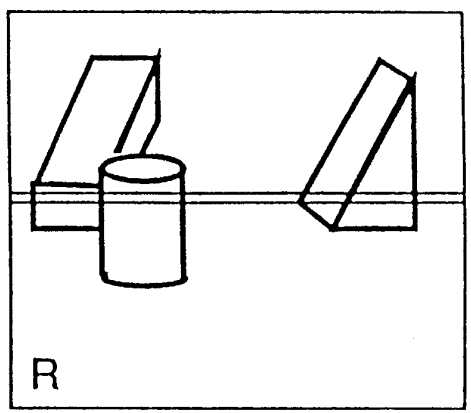
What is the thalamus doing with all that feedback?

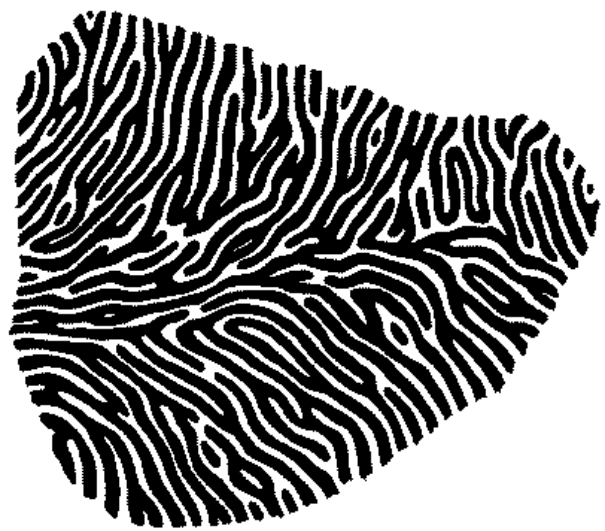
- The projections to each visual cortex first pass to the 6-layered lateral geniculate nucleus (LGN), in a polysensory organ of the midbrain called the thalamus.
- It is an intriguing fact that this "relay station" actually receives three times more descending (efferent) fibres projecting back down from the cortex, as it gets ascending (afferent) fibres from the eyes.
- Could it be that this signal 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? We will return to this theory later.
- 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 occuring here in the thalamus via this cortical feedback loop).

Interweaving data from the two eyes for stereo vision

- The right-eye and left-eye innervations from each LGN to the primary visual cortex in the occipital lobe of that hemisphere are interwoven into "slabs," or columns, in which neurones receive input primarily from just one of the eyes. Right and left eyes alternate.
- These ocular dominance columns have a cycle of about 1 mm and resemble fingerprints in scale and flow (see radiograph below).
- Clearly each hemisphere is trying to integrate together the signals from both eyes in a way suitable for stereoscopic vision, computing the relative retinal disparities of corresponding points in the images.
- The disparities reflect the relative positions of the points in depth, as we will study later in stereo algorithms.

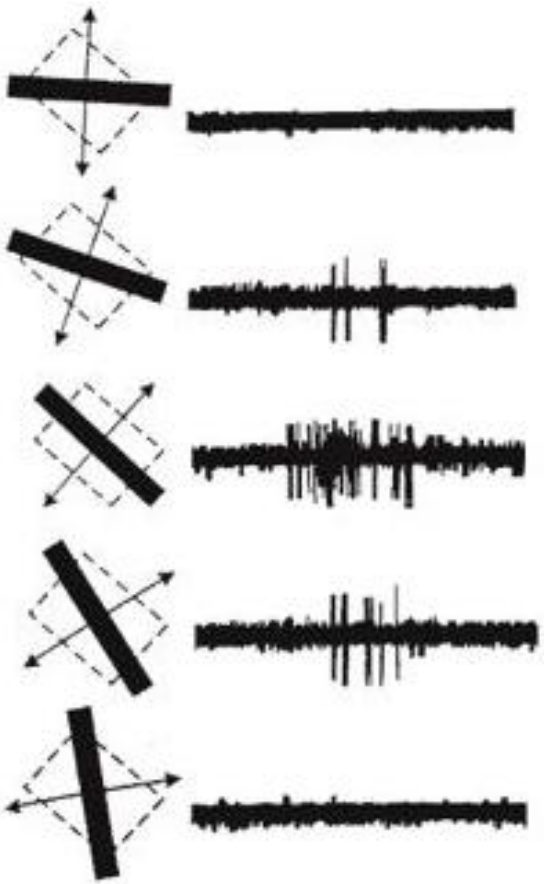


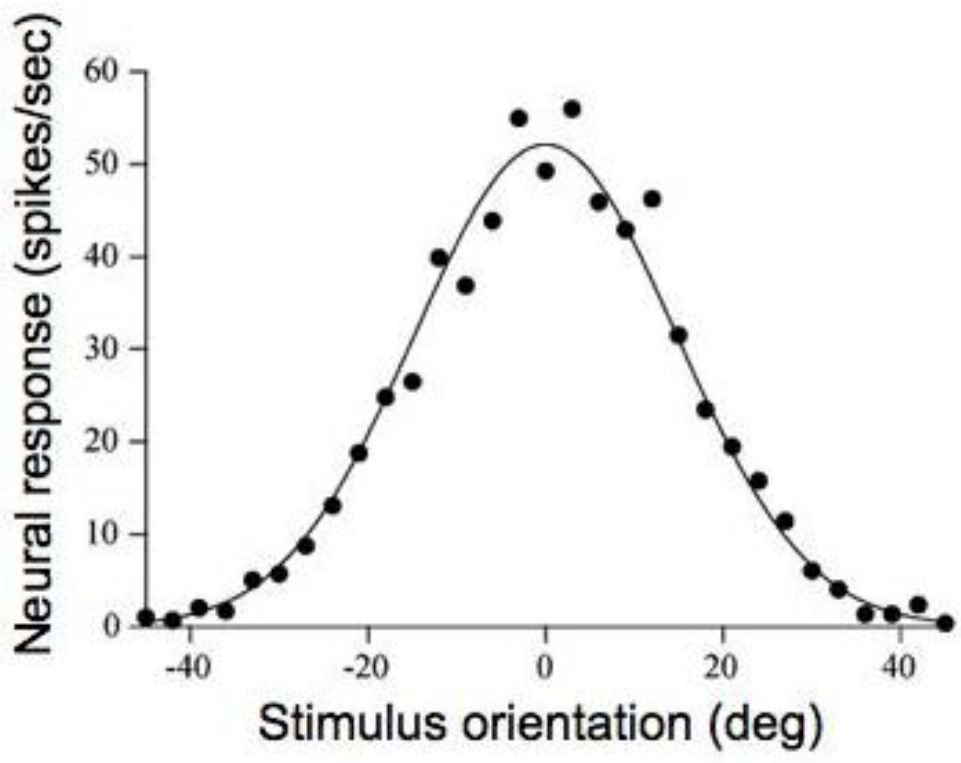




New tuning variable in visual cortex: orientation selectivity

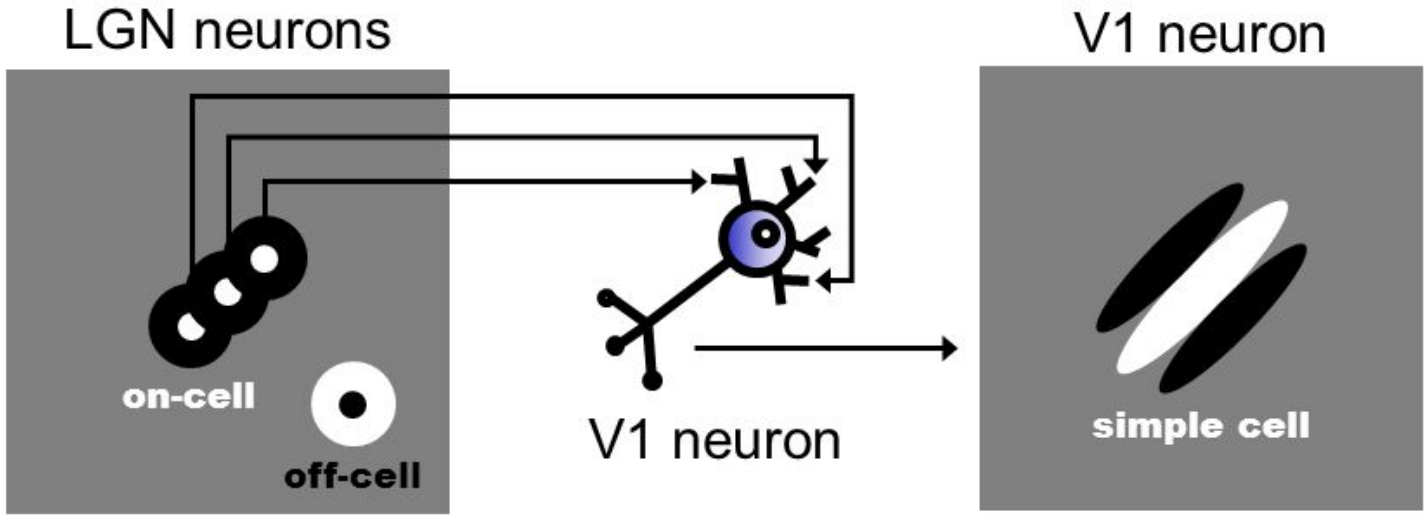
- Orthogonally to the ocular dominance columns in the visual cortical architecture, there runs a finer scale sequence of orientation columns.
- Neurones in each such column respond only to image structures (such as bars or edges) in a certain preferred range of orientations.
- Their firing rates (plotted as nerve impulses per second) reveal their selectivity for stimulus orientation: a "tuning curve":





Origin of cortical orientation selectivity

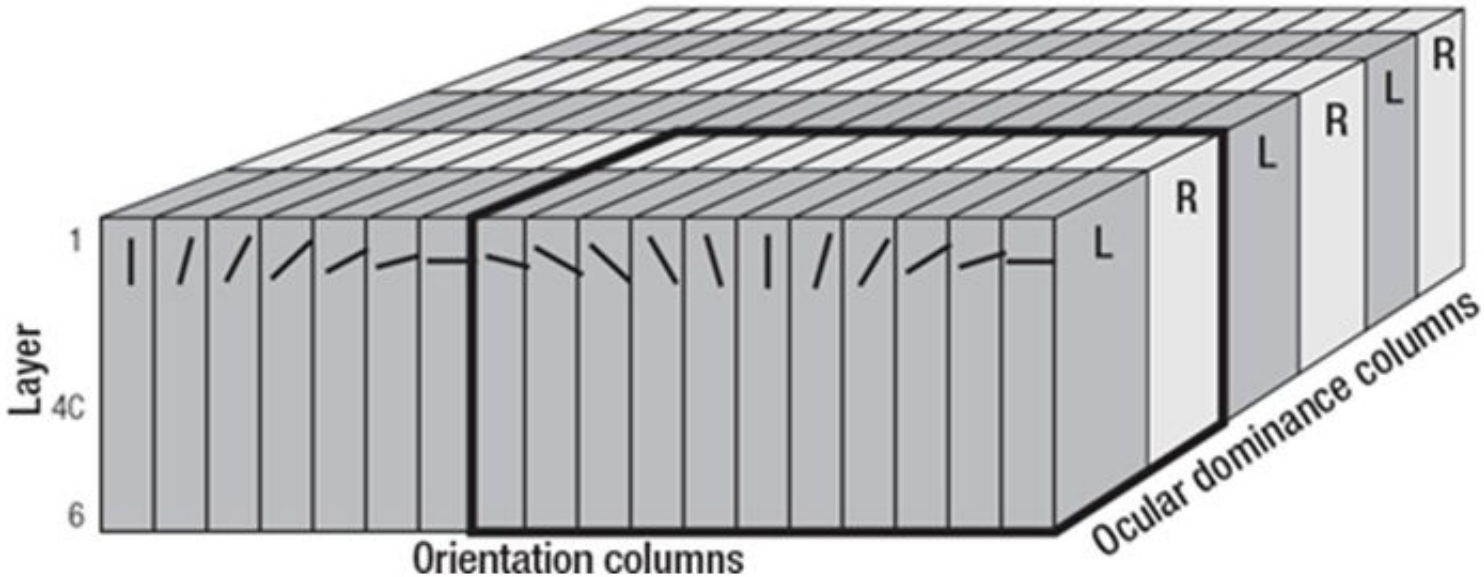
- Orientation selectivity might arise from the alignment of isotropic subunits in the LGN, summated together in their projection to the primary visual cortex (V1). Both "on" and "off" polarities exist.



- Orientation columns form a regular sequence of systematically changing preferred orientations. This sequence regularity is one of the most crystalline properties seen in visual cortical architecture.

"Hypercolumns"

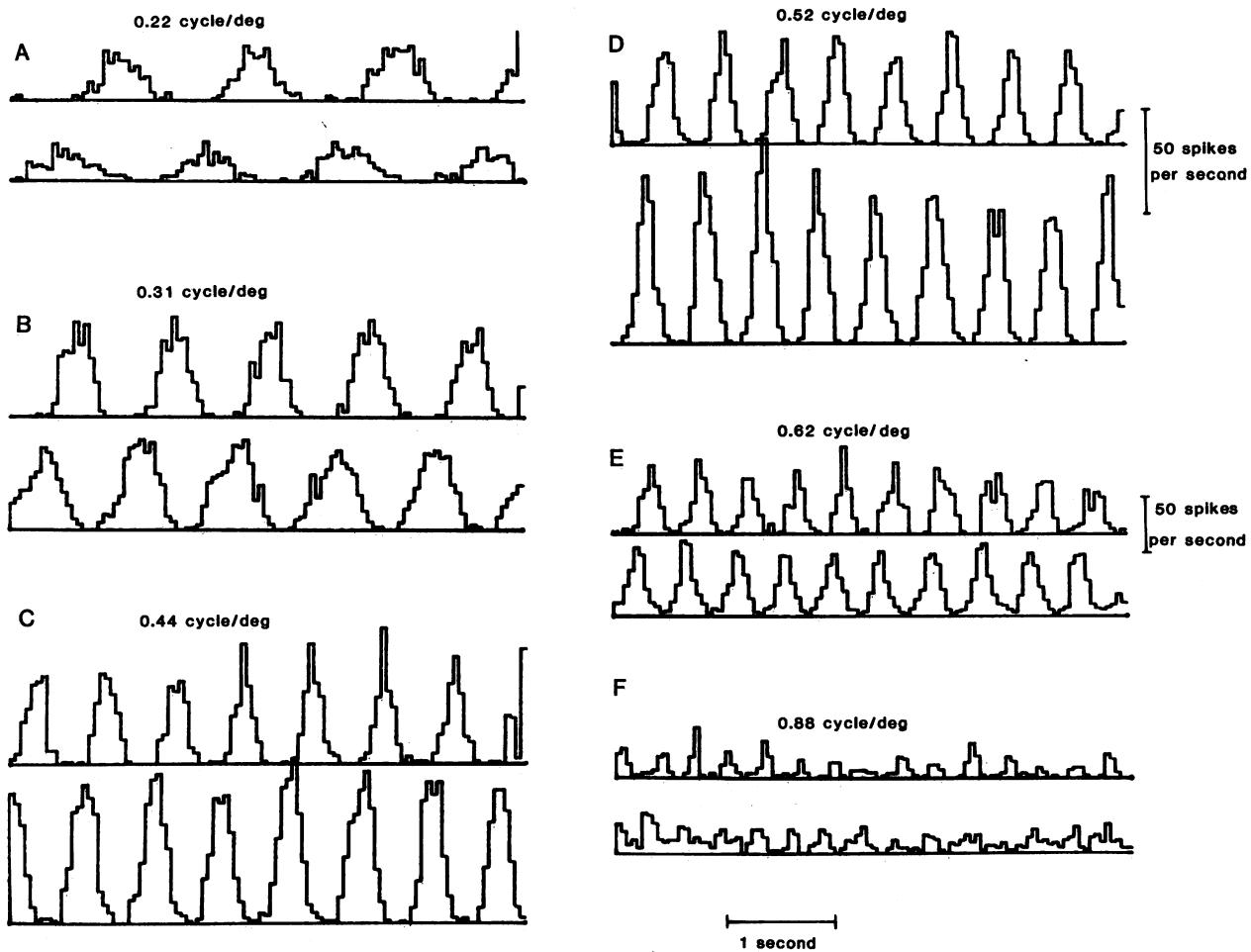
- A 3D block of about 100,000 cortical V1 neurones that includes one "right/left" cycle of ocular dominance columns, and (orthogonally organised) about ten orientation columns spanning 360° of their preferred orientations in discrete steps, is called a "hypercolumn".



- In the third dimension going down, there are six layers in which neurones vary mainly in the sizes of their receptive fields.
- This block occupies approximately 2 mm^3 of cortical tissue, and it contains the "neural machinery" to process about a 1° patch in the foveal area of visual space, or about a 6° patch in the periphery.

Quadrature phase relationships among paired V1 neurones

- Recording from adjacent pairs of neurones simultaneously, using a kind of "double-barrelled" micro-electrode, showed that neurones with the same receptive field position, orientation preference, and size, were often in quadrature phase (had a 90° spatial phase offset) when responding to a drifting sinusoidal luminance grating.



Summary of spatial image encoding in primary visual cortex

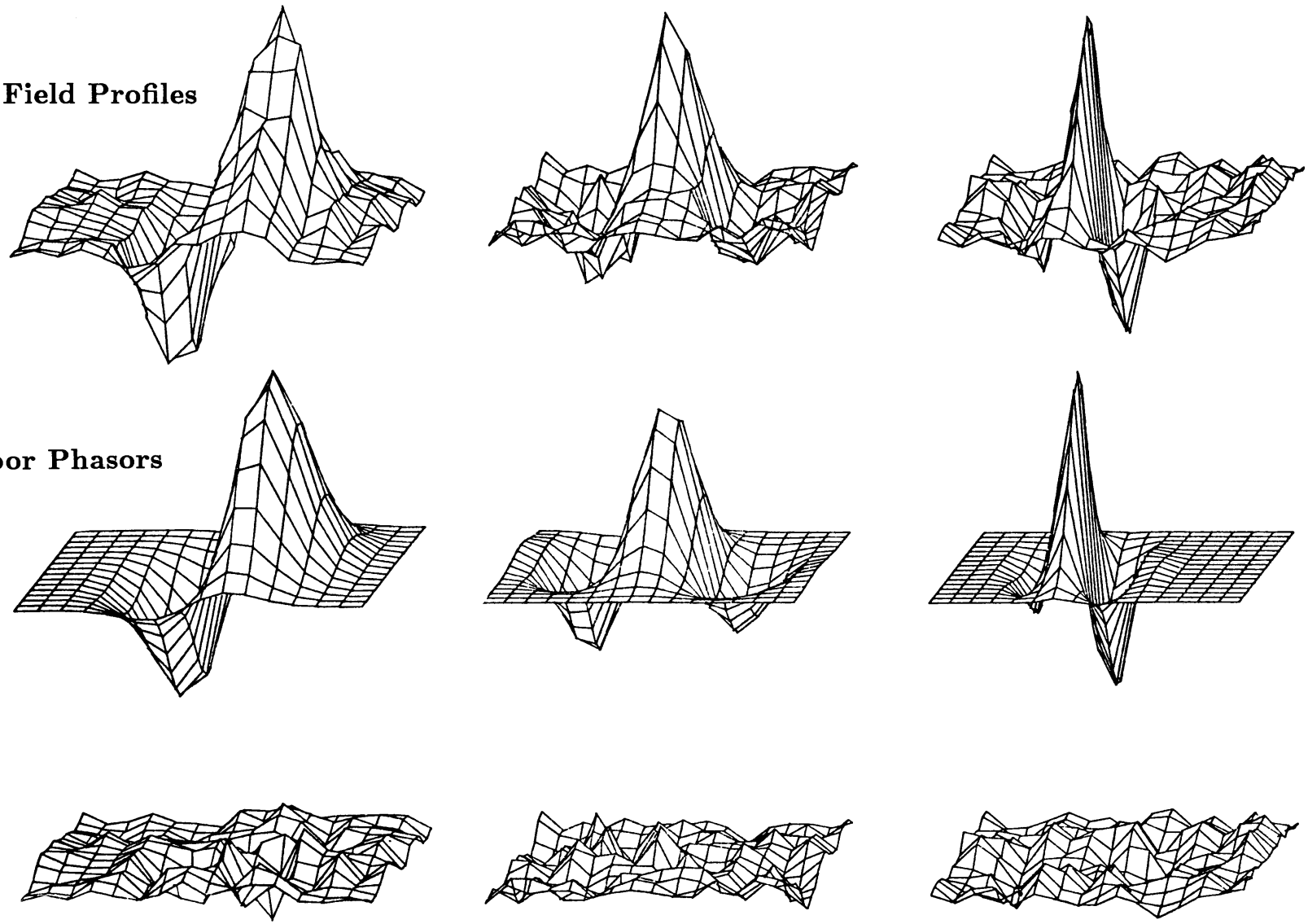
- There seem to be five main "degrees of freedom" in the spatial structure of cortical receptive field profiles: position in visual space (two coordinates), orientation preference, receptive field size, and phase (even or odd symmetry).
- These parameters can be infered from the boundaries between the excitatory and inhibitory regions, usually either bipartite or tripartite.
- Plotting how much a neurone is excited or inhibited by light as a detailed function of stimulus coordinates within its receptive field, extracts its 2D receptive field profile.
- For about 97% of such neurones studied, these receptive field profiles could be well described as 2D Gabor wavelets (or phasors).
- In the next slide, several examples of empirically measured profiles are shown in the top row; an ideal theoretical form of each such 2D Gabor wavelet (to be defined later) is shown in the middle row; and the difference between these two functions in the bottom row.
- The differences are statistically insignificant. So, it seems the brain's visual cortex discovered during its evolution the valuable properties of such 2D wavelets for purposes of image coding and analysis!

Cortical encoding of image structure by 2D Gabor wavelets

2D Receptive Field Profiles

Fitted 2D Gabor Phasors

Residuals



Historical comment

The discoveries about neurobiological mechanisms summarised above relate to six Nobel prizes won in the latter half of the 20th century:

George Wald discovered the basic photochemical reaction underlying visual transduction: 11-cis-retinal + $\hbar\nu \rightarrow$ all-trans-retinal and determined the absorption spectra of photoreceptors.

Alan Hodgkin discovered biophysical mechanisms of excitable neural membranes and how nerve impulses are generated.

Andrew Huxley discovered biophysical mechanisms of excitable neural membranes and how nerve impulses are generated.

David Hubel discovered the selectivity properties of visual neurones, and functional architecture of the visual cortex.

Torsten Wiesel discovered the selectivity properties of visual neurones, and functional architecture of the visual cortex.

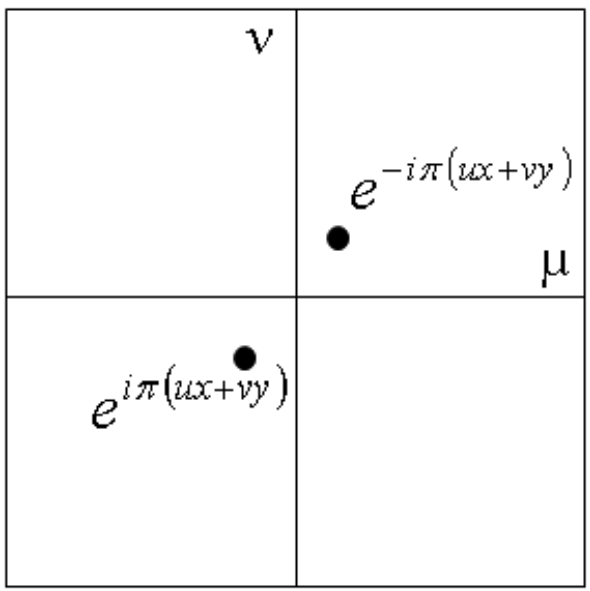
Dennis Gabor discovered the uncertainty-minimising elementary functions which we now call wavelets, and he invented the hologram.

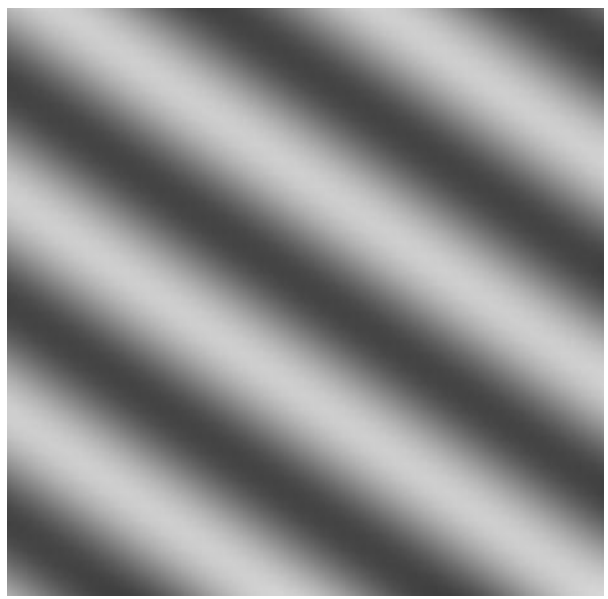
4. Mathematical image operations

- Almost all image processing begins with (2D) convolutions of an image with small kernel arrays designed for specific purposes.
- Examples include: edge detection, filtering, feature extraction, motion detection, keypoint identification, texture classification,...
- Conceptual unity: convolution $\Leftrightarrow$ filtering $\Leftrightarrow$ Fourier operation
- Even differential operators, such as taking derivatives to find edges, are implemented as convolutions with simple Fourier interpretations.
- Example: applying the Laplacian operator (sum of the image second derivatives in the vertical and horizontal directions) is equivalent to simply multiplying the Fourier transform of the image by an isotropic paraboloid: it is just a type of high-pass filtering.
- Equivalence between convolutions and (computationally simpler) Fourier domain operations make it faster to perform convolutions in the Fourier domain if the kernel chosen for the purpose is larger than (5×5), because of the huge efficiency of the Fast Fourier Transform (FFT) and the fact that convolution is replaced by multiplication.

The Fourier perspective on images

- It is therefore useful to regard an image as a superposition of many 2D Fourier components, which are complex exponential plane-waves having the form: $f(x, y) = e^{i\pi(\mu x + \nu y)}$ with complex coefficients.
- Their parameters (μ, ν) can be interpreted as 2D spatial frequency $\sqrt{\mu^2 + \nu^2}$ and orientation $\tan^{-1}(\nu/\mu)$ of the plane-wave.
- Adding together a conjugate pair of them makes a real-valued wave.
- Different images simply have different amplitudes (contrasts) and phases associated with the same universal set of Fourier components.
- Convolutions (filtering operations) just manipulate those amplitudes and phases, as a function of 2D spatial frequency and orientation.





Convolution Theorem for two-dimensional functions

Let function $f(x, y)$ have 2D Fourier Transform (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 2D integral of the product of the functions f and g after one is flipped and undergoes a relative shift by amount (x, y):

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

Thus, convolution is a way of combining two functions, making all possible relative shifts between them and integrating each such product.

The Convolution Theorem states that convolving two functions $f(x, y)$ and $g(x, y)$ together (for us usually an image and a kernel) is equivalent to just multiplying their two 2DFTs together in the 2D Fourier domain:

$$H(\mu, \nu) = F(\mu, \nu) G(\mu, \nu)$$

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

Utility of the 2D Convolution Theorem

- Nearly all feature extraction and image understanding operations begin with applying some set of filters $g_k(x, y)$ to an image $f(x, y)$, which is a linear operation implemented by convolution of $f(x, y)$ with the filter kernels $g_k(x, y)$.
- The resulting output "image" $h_k(x, y)$ then is usually subjected to non-linear operations of various kinds for analysis, segmentation, pattern recognition, and object classification.
- The Convolution Theorem is extremely useful as it is much easier simply to multiply two functions $F(\mu, \nu)$ and $G(\mu, \nu)$ together to obtain $H(\mu, \nu)$, than it is to explicitly convolve $f(x, y)$ and $g(x, y)$ together (if the kernel is larger than about 5 x 5) to obtain $h(x, y)$.
- Of course, exploiting the Convolution Theorem means going into the 2D Fourier Domain first and computing the 2DFT's of $f(x, y)$ and $g(x, y)$, and then performing yet another (inverse) Fourier transform in order to recover $h(x, y)$ from the resulting $H(\mu, \nu)$. But with the powerful 2D-FFT algorithm, this is very efficient. Moreover, the 2DFTs of the filter kernels $g_k(x, y)$ are usually known in advance.

Pseudo-code for explicit image convolution with a kernel

The discrete convolution of an image array with a 2D filter kernel can be represented algebraically as follows, where the continuous integrals are replaced by discrete summations (with array boundary issues ignored). Note that there are four nested for-loops:

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

```
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);
                                    }
                                }
```

Comparative analysis of computational efficiency

- If we implement the convolution in the Fourier domain because the kernel array was large, then of the four nested **for loops**, the inner two **for loops** would be entirely eliminated.
- Instead, the only operation inside the outer two **for loops** would be:

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

- ...but first we need FFTs of the kernel (trivial) and of the image.
- 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 two nested **for loops**, the total complexity of the 2D Fourier approach is $n^2(2\log_2(n) + 1)$.
- In contrast, the number of multiplications for explicit convolution 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, the 2D Fourier approach is faster.

Differentiation Theorem

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

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

Thus calculus gets replaced by algebra, in the Fourier domain.

A particularly useful implication is that isotropic differentiation, which treats all directions equally (and 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:

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

Practical Application: Multi-Resolution Edge Detection

5. Edge detection

Whether edges are straight, curved, or forming closed boundary contours, they are very informative for several reasons:

- Edges demarcate the boundaries of objects, or of material properties.
- Objects have parts, which typically make edges where they join.
- 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 constitutes an "edge" which is very useful for that domain.
- Aligning edges is a way to solve the stereo correspondence problem.
- A correspondence problem exists also for frames displaced in time.
- Velocity fields, containing information about object trajectories, can be organized and understood by the movements of edges. Motions of objects generate velocity discontinuities at their boundaries.

In summary, DISCONTINUITIES = INFORMATION.

Differential operators for edge detection

Edges mark loci of sudden change, so a natural approach to detecting them is to estimate (directional) derivatives across an image frame.

Recall the "Newton Quotient" defining the first derivative of $f(x)$:

$$\frac{df(x)}{dx} \equiv \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

or for a 2D function $f(x, y)$ such as an image, its partial derivatives:

$$\frac{\partial f(x, y)}{\partial x} \equiv \lim_{\Delta x \to 0} \frac{f([x + \Delta x], y) - f(x, y)}{\Delta x}$$

$$\frac{\partial f(x, y)}{\partial y} \equiv \lim_{\Delta y \to 0} \frac{f(x, [y + \Delta y]) - f(x, y)}{\Delta y}$$

These let us define the gradient vector field $\vec{\nabla} f(x, y)$ over the image:

$$\vec{\nabla} f(x, y) \equiv \left(\frac{\partial f(x, y)}{\partial x}, \frac{\partial f(x, y)}{\partial y} \right)$$

Finite difference approximations as discrete derivatives

For a discrete image array $f[n, m]$, clearly $\Delta x, \Delta y$ are fixed at pixel size.

The discrete form of a first derivative is a finite difference which simply subtracts adjacent values, by row or by column: $f[n+1, m] - f[n, m]$ or $f[n, m+1] - f[n, m]$. These generate entire new images.

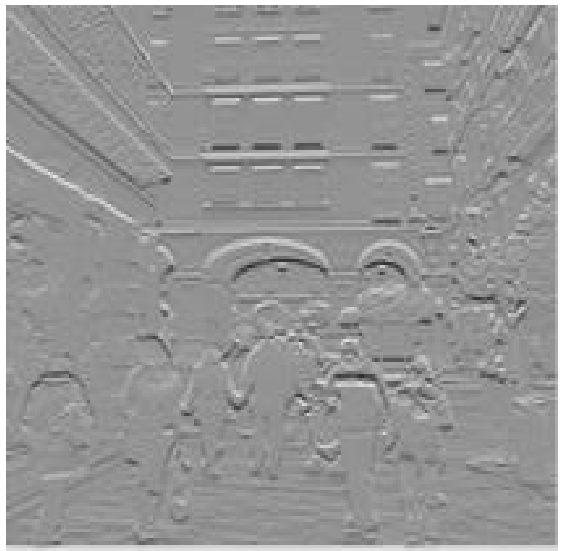
Computing these new images can be achieved by the discrete convolution of our image array $f[n, m]$ with the first finite difference kernel $[-1, 1]$ (for vertical edges) or with $[-1, 1]^{\mathrm{T}}$ (for horizontal edges). These kernels should be regarded as arrays, either (1×2) or (2×1).

The next slide illustrates their effects on an actual greyscale image, to pick out either the vertical or the horizontal edges. But note these edge detectors are also polarity sensitive: they distinguish between a "rightward edge" (say from dark to bright) versus a "leftward edge" (bright to dark). We might prefer polarity-independent edge detection.

First finite difference operators for detecting edges

convolution with $[-1, 1]$ $\Longleftarrow$ ORIGINAL $\Longrightarrow$ convolution with $[-1, 1]^{\mathrm{T}}$





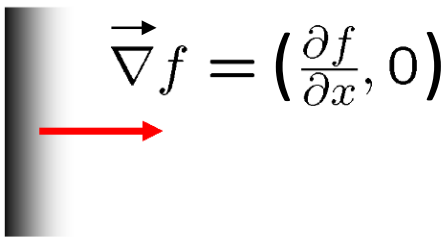
Successive concatenations of first finite difference kernels generate the discrete approximations to successively higher derivatives, as you might expect: $[-1, +2, -1]$, $[-1, +3, -3, +1]$, $[-1, +4, -6, +4, -1]$, etc. But the higher derivatives become successively less useful because they become increasingly noisy. Recall from the Differentiation Theorem that they amount to high-pass filtering with successive powers of frequency, thus greatly increasing the high-frequency noise.

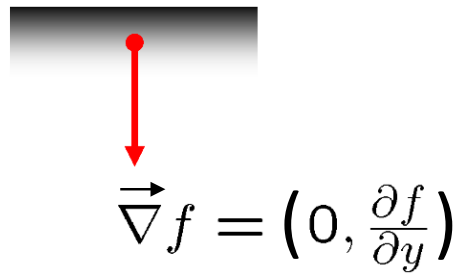
Gradient vector field: both edge magnitude and direction

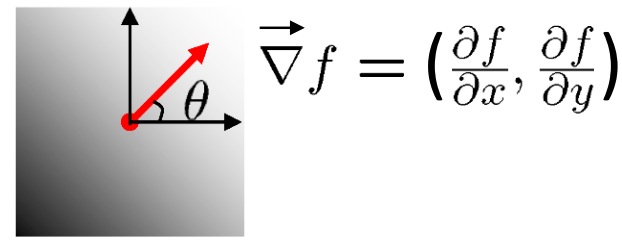
The gradient vector field over an image, as introduced earlier:

$$\vec{\nabla} f(x,y) \equiv \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

locally points in the direction of most rapid change of image intensity:







The gradient direction (orientation of an edge normal) is given by:

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

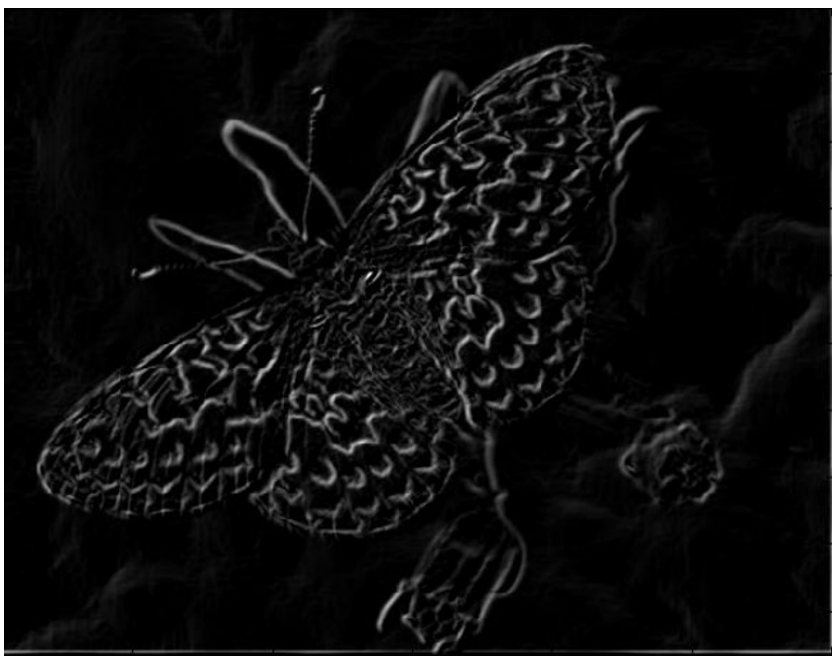
while the edge strength is given by the gradient magnitude:

$$\|\vec{\nabla} f\| = \sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2}$$

Gradient magnitude

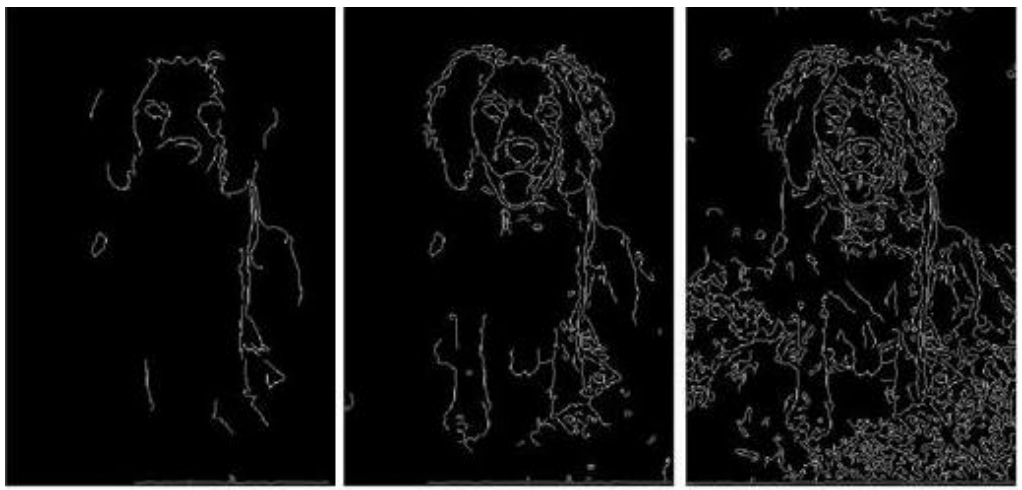
Gradient magnitude usefully both detects edges and shows their strength:

$$\|\vec{\nabla} f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

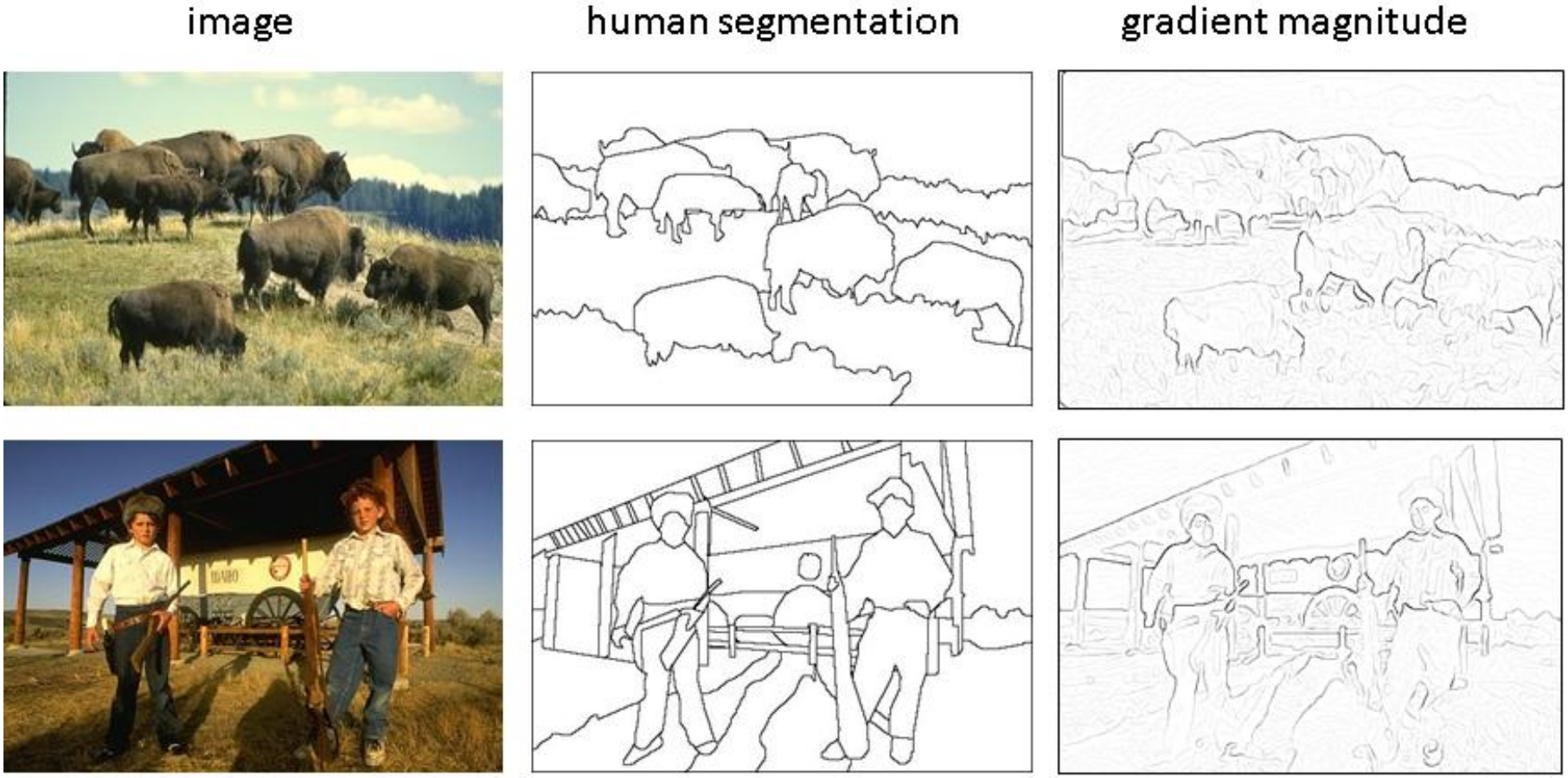


Problem with noise and clutter in edge detection

Unfortunately, object boundaries of interest are sometimes fragmented, and spurious "clutter" edge points are found. These problems are not solved, but traded-off, by applying a threshold to the gradient magnitude:



Humans perform better at image segmentation



Combining 2nd-order differential operators with smoothing

An alternative to the gradient vector field is a second-order differential operator, combined with smoothing at a specific scale of analysis. An example of a 2D kernel based on the second finite difference operator is:

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

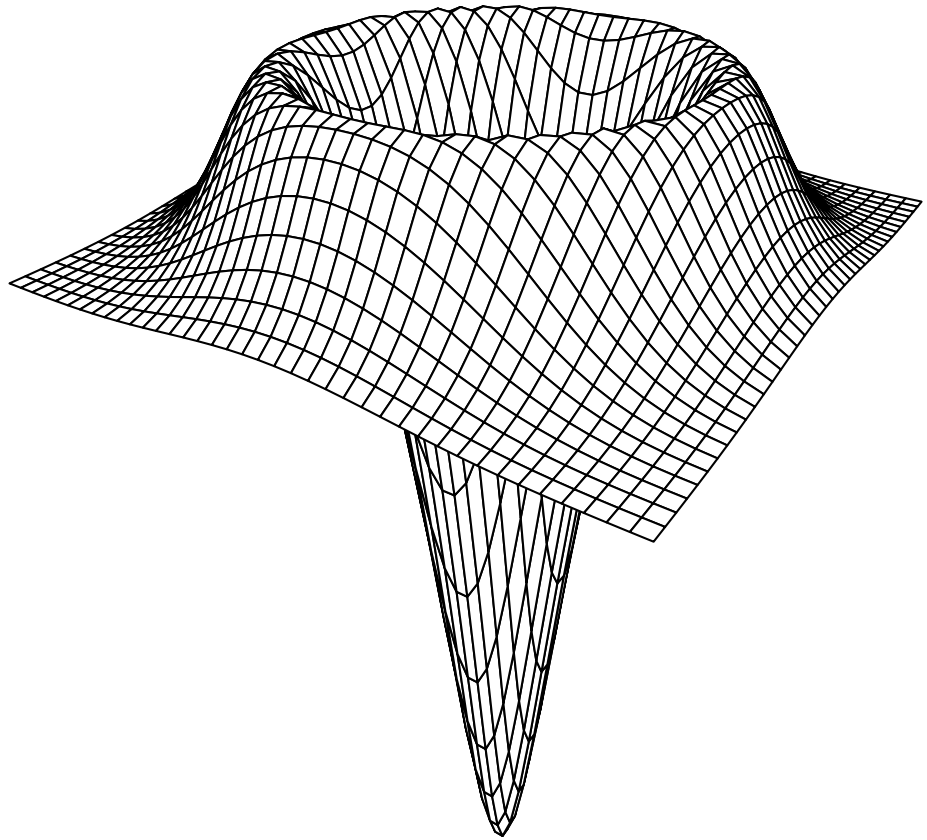
Clearly, such an operator will detect edges only in a specific orientation. It is integrating in the vertical direction, and taking a second derivative horizontally. In comparison, an isotropic operator such as the Laplacian (sum of 2nd derivatives in two orthogonal orientations) has no preferred orientation; that is the meaning of isotropy. A discrete approximation to the Laplacian operator ∇^2 (no smoothing) in just a small (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. Therefore they give no response to uniform illumination, but respond only to actual image structure.

Scale-specific edge operator: Laplacian of a Gaussian

A popular second-order differential operator for detecting edges at a specific scale of analysis, with a smoothing parameter σ, is $\nabla^2 G_\sigma(x, y)$:



For a parameterised Gaussian form $G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$, we have

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

Why specify a scale of analysis for edge detection?

- Edges in images are defined at different scales: some transitions in brightness are gradual, others very crisp. Importantly, at different scales of analysis, different edge structure emerges.
- Example: an image of a leopard that has been low-pass filtered (or analyzed at a coarse scale) has edge outlines corresponding to the overall form of its body.
- At a somewhat finer scale of analysis, image structure may be dominated by the contours of its "spots." At a still finer scale, the relevant edge structure arises from the texture of its fur.
- In summary, non-redundant structure exists in images at different scales of analysis (or if you prefer, in different frequency bands).
- The basic recipe for extracting edge information from images is to use a multi-scale family of filters as the image convolution kernels.
- One approach is to apply a single filter to successively downsampled copies of the original image. A Laplacian pyrammid thereby extracts image structure in successive octave bands of spatial frequencies.

Different image structure at different scales of analysis

Video demonstration of edge information in the high spatial frequencies:

https://www.youtube.com/watch?v=2rDRAa14KdU

Concatenation of smoothing and differentiation steps

- In the 2D Fourier domain, as we have seen, the spectral consequence of applying the Laplacian operator $\nabla^2 \equiv \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right)$ to an image is to multiply the image spectrum by a paraboloid: $(\mu^2 + \nu^2)$.
- Clearly this emphasizes the high frequencies at the expense of the low frequencies, and it eliminates the "DC" component entirely (hence the output image has average "brightness" 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. 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 the scale parameter σ is used.
- The zero-crossings of the (zero-centred) output image correspond to edge locations. Thus edges are detected with invariance to polarity.
- The relevance of Logan's Theorem (richness of zero-crossings in one-octave bandpass signals) is asserted in favour of $\nabla^2 G_\sigma(x, y)$, although 1.3 octaves doesn't quite satisfy the one-octave constraint.

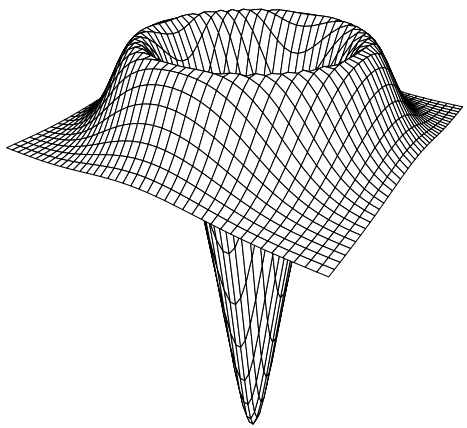
(Concatenation of smoothing and differentiation, con't)

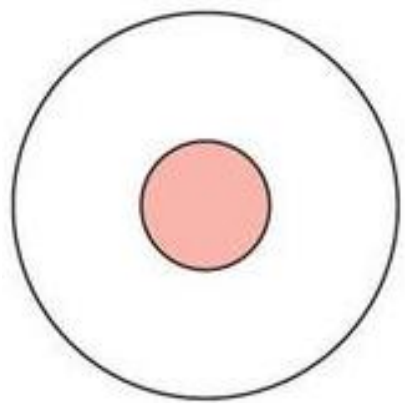
- Note that by commutativity of linear operators, the order in which these steps are applied to the image $I(x, y)$ doesn't matter. First computing the Laplacian of the image, and then blurring the result with the Gaussian, is equivalent to first convolving the image with the Gaussian and then computing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

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

- Both of these sequences are equivalent to just convolving the image once 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.

- The filter kernel $\nabla^2 G_\sigma(x, y)$ *does* bear a noteworthy resemblance to those "on/off centre-surround" neural receptive field profiles:





Canny edge operator

A computationally more complex approach to edge detection was developed by Canny, to avoid the spurious edge clutter seen earlier. It is popular because it is better able to distinguish real edges that correspond to actual object boundaries.

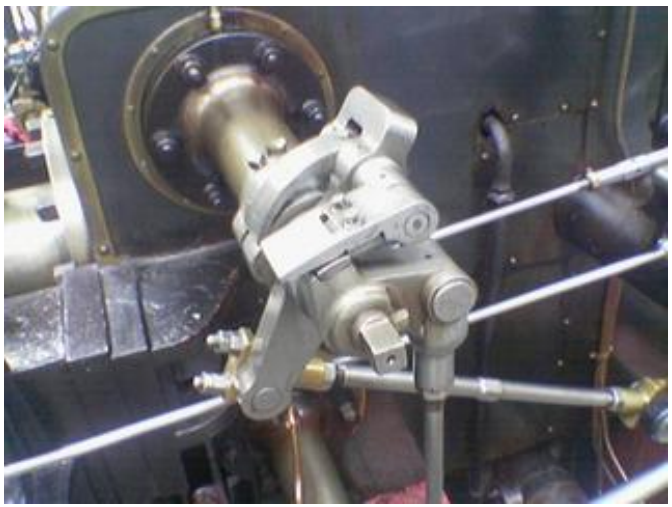
The Canny edge operator has five main steps (two discussed earlier):

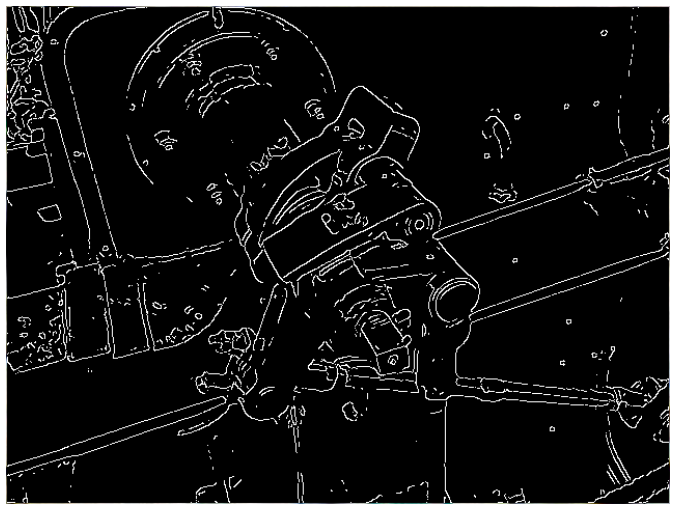
1. Smooth the image with a Gaussian filter to reduce noise.
2. Compute the gradient vector field $\vec{\nabla} I(x, y)$ over the image.
3. Apply an "edge thinning" technique, non-maximum suppression, to eliminate spurious edges. A given edge should be represented by a single point, at which the gradient is maximal.
4. Apply a double threshold to the local gradient magnitude, resulting in three classes of edge data, labelled strong, weak, or suppressed. The threshold values are adaptively determined for a given image.
5. Impose a connectivity constraint: edges are "tracked" across the image; edges that are weak and not connected to strong edges are eliminated.

Cleaner results using the Canny edge detector

input image

output edge map





The resulting edge map corresponds better to actual object boundaries, and the computed edges are marked as only one pixel large.

6. Multiscale wavelets for image analysis; active contours

An effective method to extract, represent, and analyse image structure is to compute its 2D Gabor wavelet coefficients.

- The 2D form of Gabor wavelets were originally proposed as a model for the receptive field profiles of neurones in the brain's visual cortex, and for understanding their selectivity properties.
- These 2D wavelets are optimal for extracting the maximum possible information about the orientation and modulation of image structure ("what"), simultaneously with 2D position ("where").
- The 2D Gabor filter family achieves the theoretical lower bound on joint uncertainty over these variables in the Uncertainty Principle.
- These properties are particularly useful for texture analysis because of the 2D spectral specificity of texture and its spatial variation.
- These wavelets are also used for motion detection, stereoscopic vision, and in many sorts of visual pattern recognition.
- A large and growing literature now exists on the efficient use of this image representation basis, and its applications in vision.

(2D Gabor wavelets, con't)

Two-dimensional Gabor wavelets have the functional form:

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

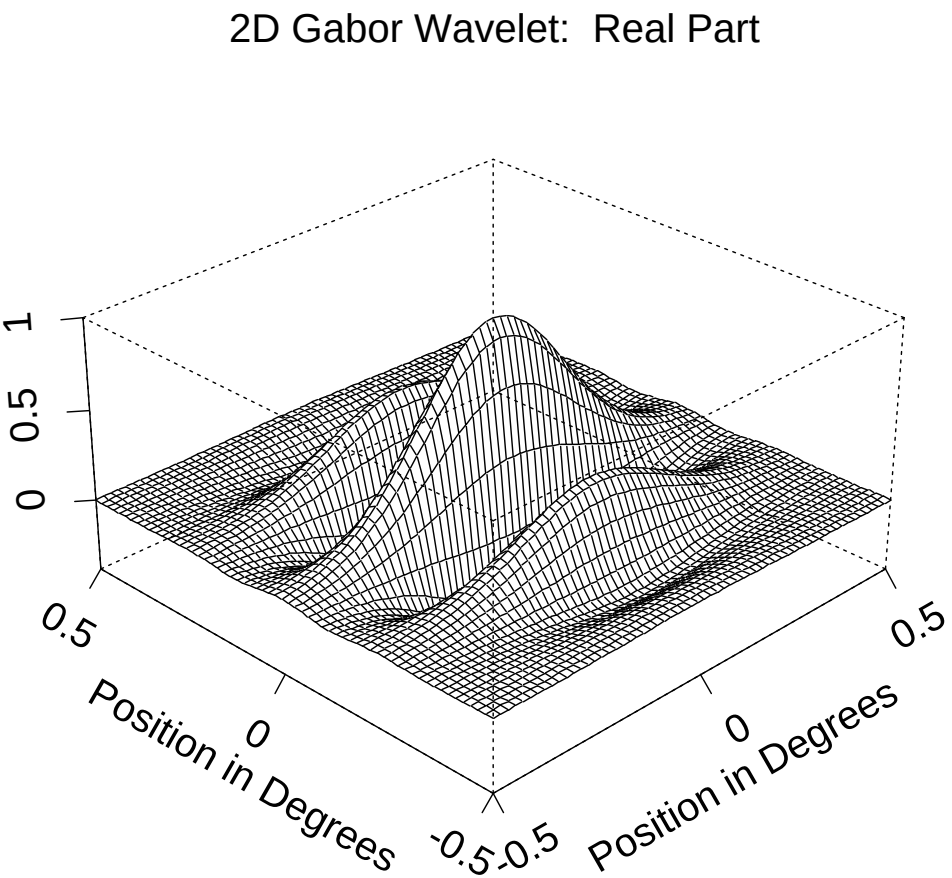
- (x_0, y_0) specify position in the image,
- (α, β) specify effective width and length,
- (u_0, v_0) specify modulation, of spatial frequency $\omega_0 = \sqrt{u_0^2 + v_0^2}$ and orientation $\theta_0 = \tan^{-1}(v_0/u_0)$.
- You may recall that these were also the main degrees-of-freedom describing neurones in the brain's primary visual cortex (V1).
- (A further degree-of-freedom not included here is the relative orientation of the elliptic Gaussian envelope, which creates cross-terms in xy.)

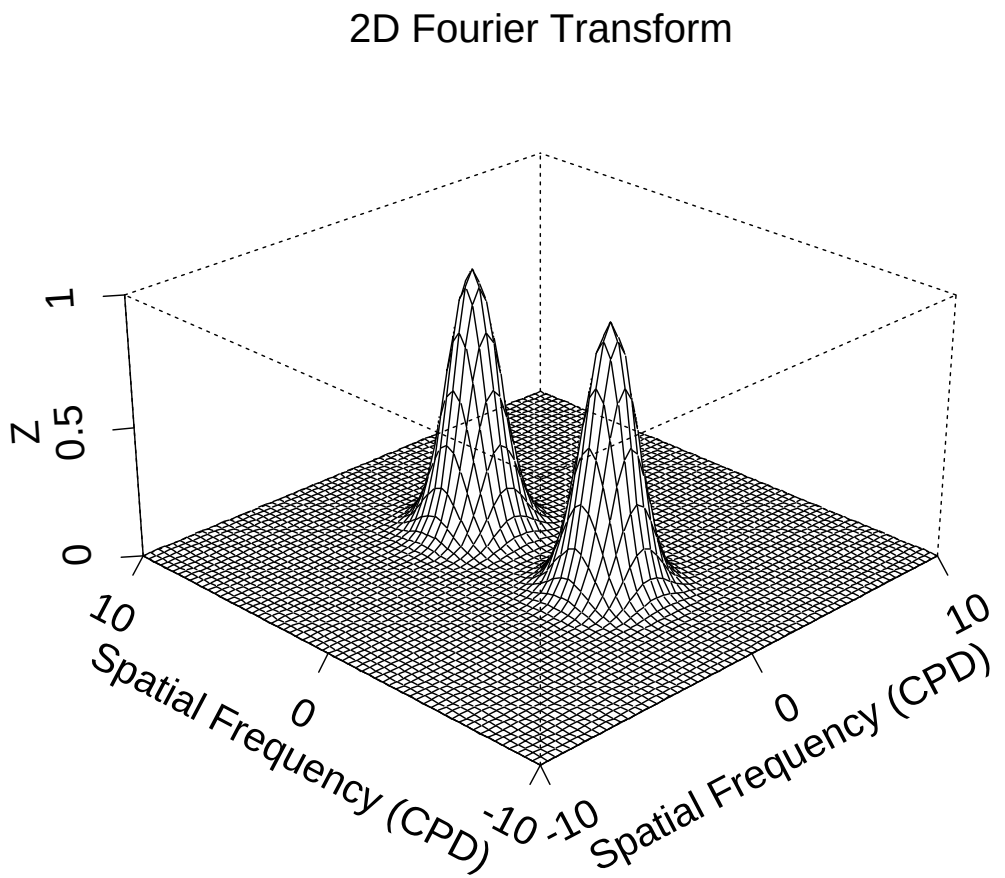
The 2D Fourier transform $F(u, v)$ of a 2D Gabor wavelet has exactly the same functional form, with parameters just interchanged or inverted:

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

(2D Gabor wavelets, con't)

The real part of one member of the 2D Gabor wavelet family, centered at the origin $(x_0, y_0) = (0, 0)$ and with unity aspect ratio $\beta/\alpha = 1$ is shown, together with its 2D Fourier transform $F(u, v)$. There are two peaks because two complex exponentials are required to make a cosine wave, which is the real part of modulation (couldn't plot a 4D surface).

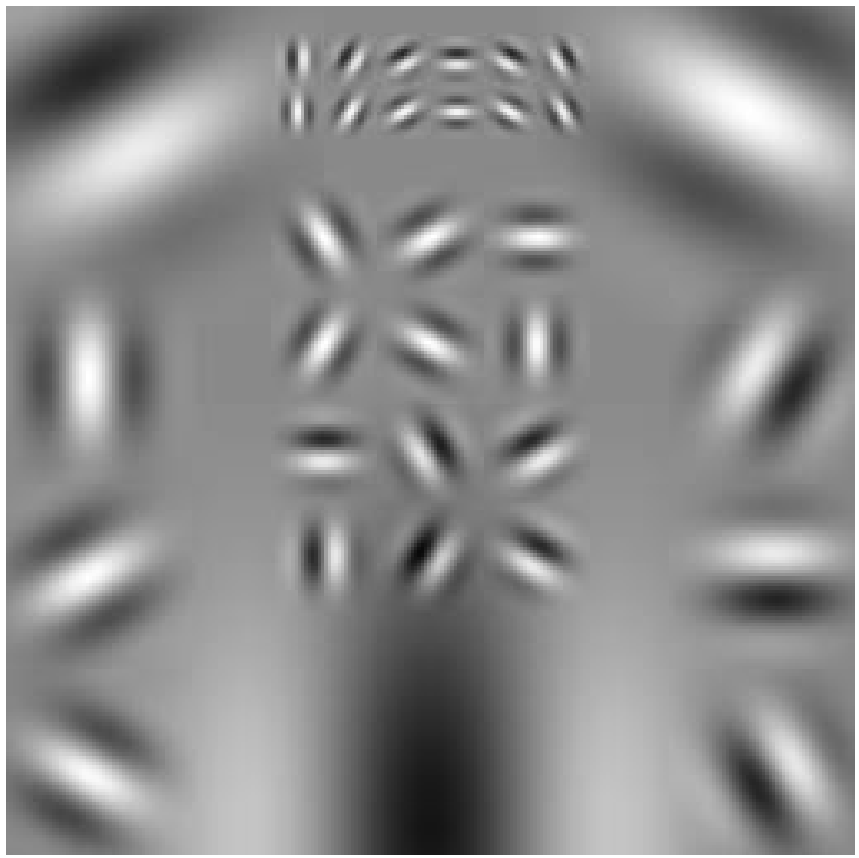




(2D Gabor wavelets, con't)

With parameterisation for dilation, rotation, and translation, such 2D wavelets can form a complete and self-similar basis for representing and analysing the structure in images.

Here are examples of a wavelet family codebook having five sizes, by factors of two (thus spanning four octaves), six orientations in 30 degree increments, and two phases, over a lattice of positions.



(2D Gabor wavelets, con't)

Self-similarity is reflected in using a generating function. If we take $\Psi(x, y)$ to be some chosen generic 2D Gabor wavelet, then we can generate from this one member, or "mother wavelet", the self-similar family of daughter 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:

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

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

These properties of self-similarity can be exploited for constructing efficient, compact, multiscale codes for image structure.

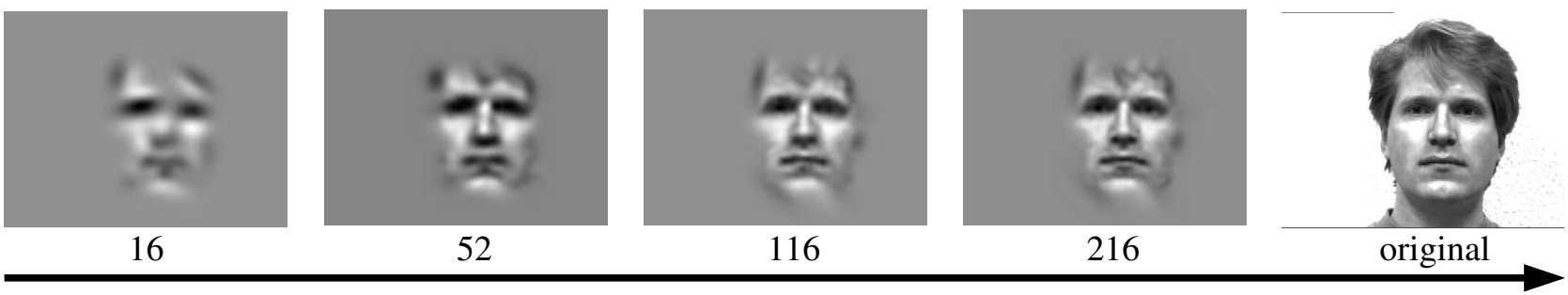
(2D Gabor wavelets, con't)

The completeness of 2D Gabor wavelets as a basis for image analysis can be shown by reconstructing a facial image from them, in stages.

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

A "philosophical" comment about Gabor wavelets

- Aristotle defined vision as "knowing what is where." We have noted the optimality of 2D Gabor wavelets for simultaneously 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. Currently this is the standard model for how the brain's visual cortex represents the information in the retinal image.
- The 2D Gabor framework has also become ubiquitous in Computer Vision, not only as the "front-end" representation but also as a general toolkit for solving many practical problems. Thus we have seen the migration of an idea from neurobiology into mainstream engineering, mathematical computing, and artificial intelligence.



Detection of facial features using quadrature wavelets

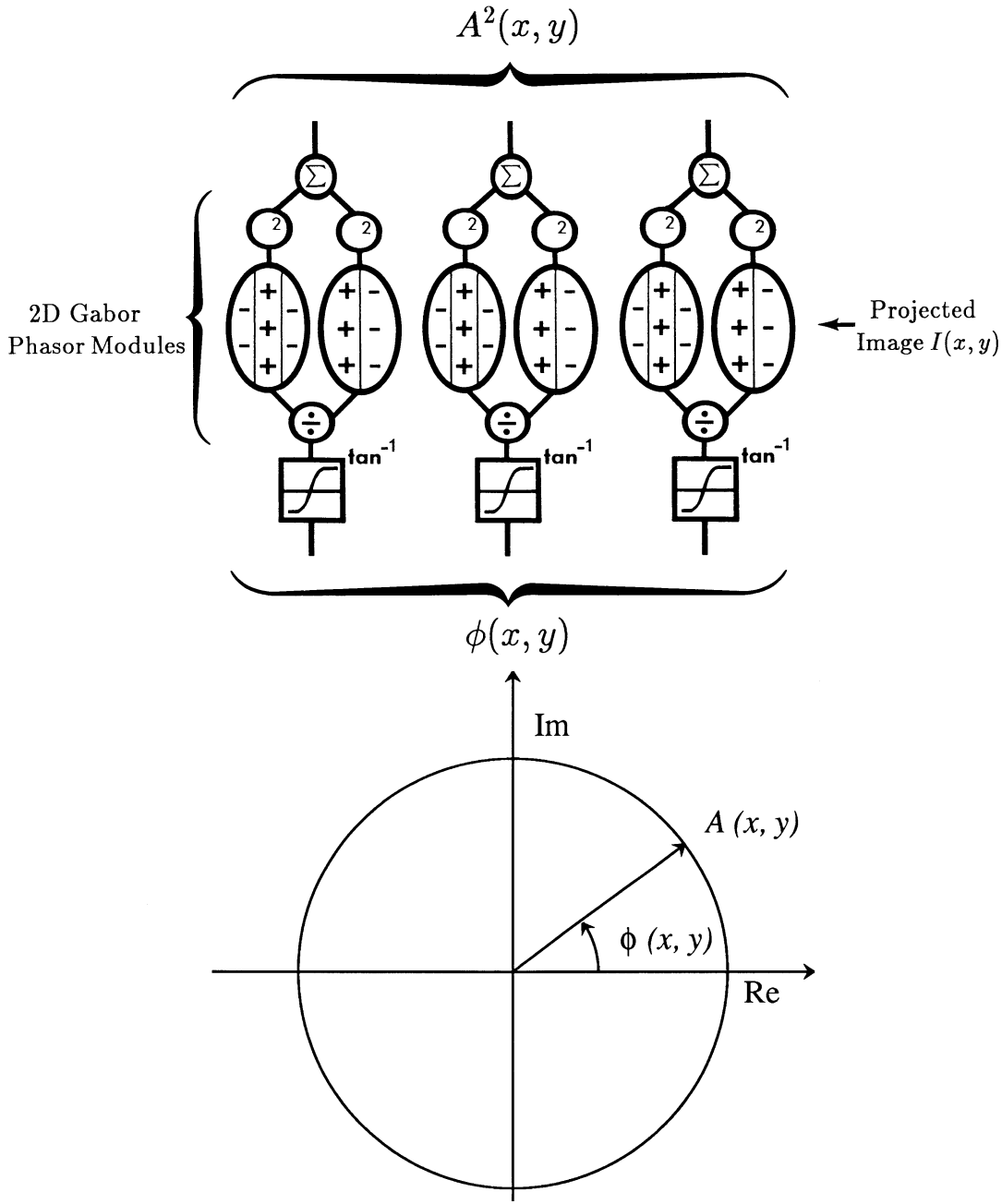
- An illustration of a practical application of such image operators is in the automatic localisation of facial features.
- Interestingly, most facial features themselves can be captured using only a handful of wavelets, since such features are (after all) just localized undulations having certain positions, spatial frequencies, orientation, and phases.
- By taking the modulus (sum of the squares of the real and imaginary parts) of a facial image after convolving it with complex-valued 2D Gabor wavelets, key facial features (eyes and mouth) are detected readily; we may call this a quadrature demodulator neural network. (Note that modulation here is in the x-direction only, for simplicity.)

$$g(x,y) = \int_{\alpha}\int_{\beta} e^{-\left((x-\alpha)^2+(y-\beta)^2\right)/\sigma^2} \cos(\omega(x-\alpha))\ I(\alpha,\beta)\ d\beta\ d\alpha$$

$$h(x,y) = \int_{\alpha}\int_{\beta} e^{-\left((x-\alpha)^2+(y-\beta)^2\right)/\sigma^2} \sin(\omega(x-\alpha))\ I(\alpha,\beta)\ d\beta\ d\alpha$$

$$A^2(x,y) = g^2(x,y) + h^2(x,y)$$

Quadrature demodulator neural network

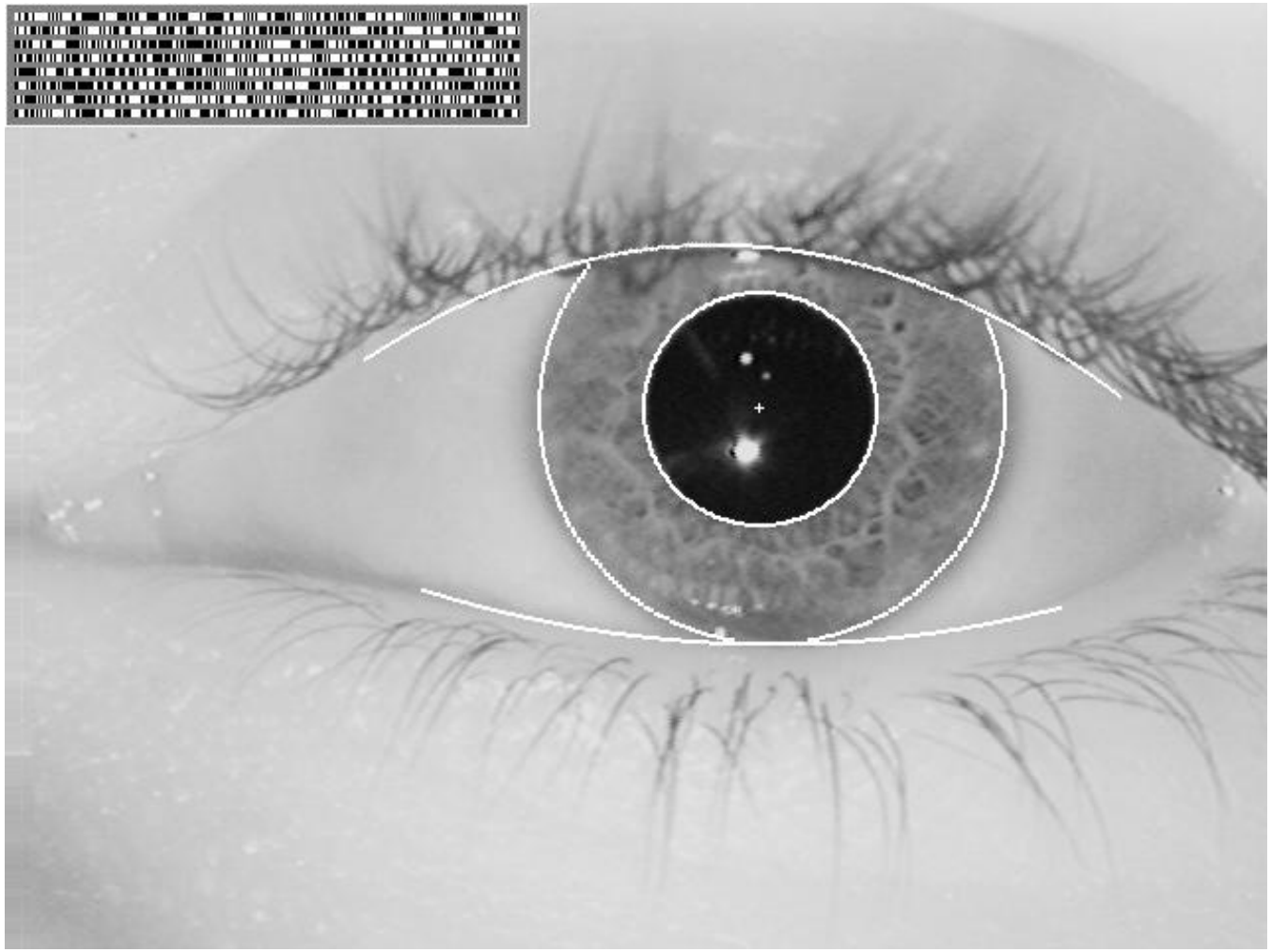


Detection of facial features using quadrature wavelets

Left panel: original image. Right panel (clockwise from top left): real part after 2D Gabor wavelet convolution; imaginary part; modulus; and modulus superimposed on the original (faint) image, illustrating feature localisation.

Edge detection and selection constrained by shape models

- Integro-differential operators for edge detection can be constrained so that they find only certain specified families of boundary shapes.
- By computing derivatives of contour integrals along shaped paths, it is possible to find (say) only circular or parabolic boundary shapes.



Parameterised edge selection by voting; Hough transform

- The white boundaries isolated in the previous slide are the curves whose parameters were found to maximise the blurred derivatives, with respect to increasing radius, of contour ("path") integrals:

$$\arg\max_{(r,x_0,y_0)} \left| G_\sigma(r) * \frac{\partial}{\partial r} \oint_{r,x_0,y_0} \frac{I(x,y)}{2\pi r} ds \right|$$

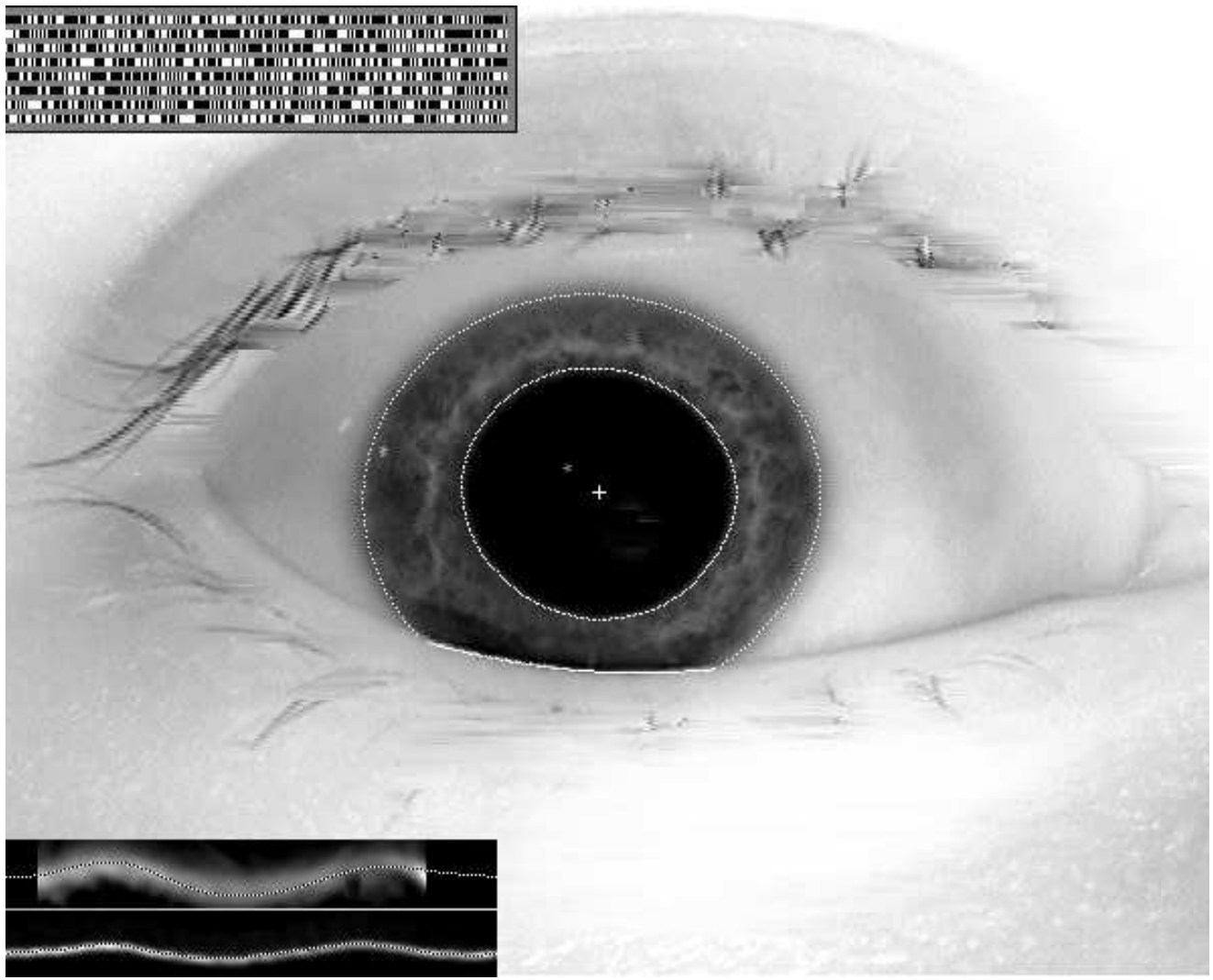
- Voting schemes such as the Hough transform seek to find instances of shapes, however imperfect, within a certain class of shapes.
- Parameters of curves (like the circles' r, x_0, y_0 in the previous slide) define an accumulator space for grouping together edge evidence.
- Edge evidence might be (qualifying) gradient magnitudes, or the output of the Canny operator with its thinning/cleaning processes.
- For each edge pixel, increment all the compatible accumulator cells.
- The accumulator cell (like parameters r, x_0, y_0) for which the greatest amount of edge evidence can be found, is the "winning" curve.

Active contours for boundary descriptors

- 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 "stiffness," 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 goals about shape, such as geometry, complexity, classification and smoothness, and also with theory of evidence and data fusion.
- The image of the eye (next slide) 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. These must be accurately localised in order for the biometric technology of *iris recognition* to work.
- Evidence about the 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.

(Active contours for boundary descriptors, con't)

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).



(Active contours for boundary descriptors, con't)

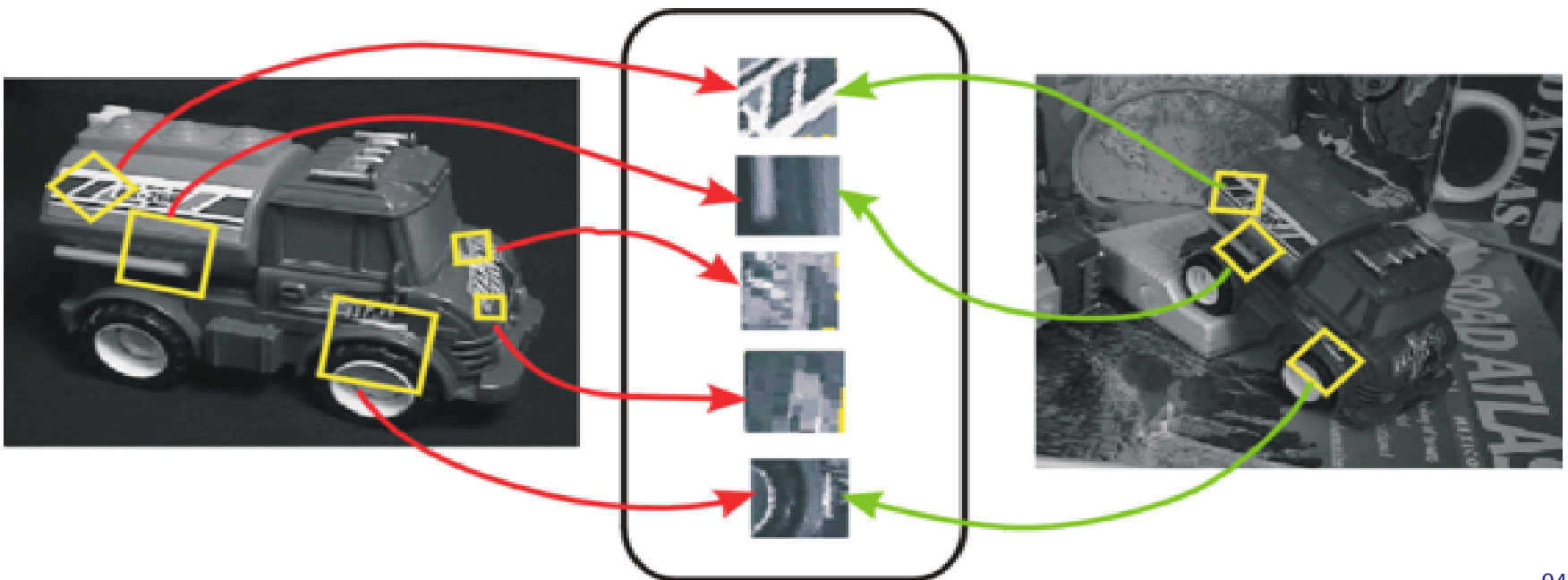
- Match a deformable model to an image, by "energy minimisation"
- Used for shape recognition, object tracking, and image segmentation
- A deformable spline (or "snake") changes its shape under competing forces: image forces that pull it towards certain object contours; and internal forces ("stiffness") that resist excessive deformations
- The trade-off between these forces is adjustable, and adaptable
- External energy reflects how poorly the snake is fitting a contour
- Internal energy reflects how much the snake is bent or stretched
- This sum of energies is minimised by methods like gradient descent, simulated annealing, and partial differential equations (PDEs)
- Problems: numerical instability, and getting stuck in local minima
- With geodesic active contours (used in medical image computing), contours may split and merge, depending on the detection of objects in the image

Demonstration: https://www.youtube.com/watch?v=ceIddPk78yA

Scale-Invariant Feature Transform (SIFT)

Goals and uses of SIFT:

- Object recognition with geometric invariance to transformations in perspective, size (distance), position, and pose angle
- Object recognition with photometric invariance to changes in imaging conditions like brightness, exposure, quality, wavelengths
- Matching corresponding parts of different images or objects
- "Stitching" overlapping images into a seamless panorama
- 3D scene understanding (despite clutter)
- Action recognition (what transformation has happened...)



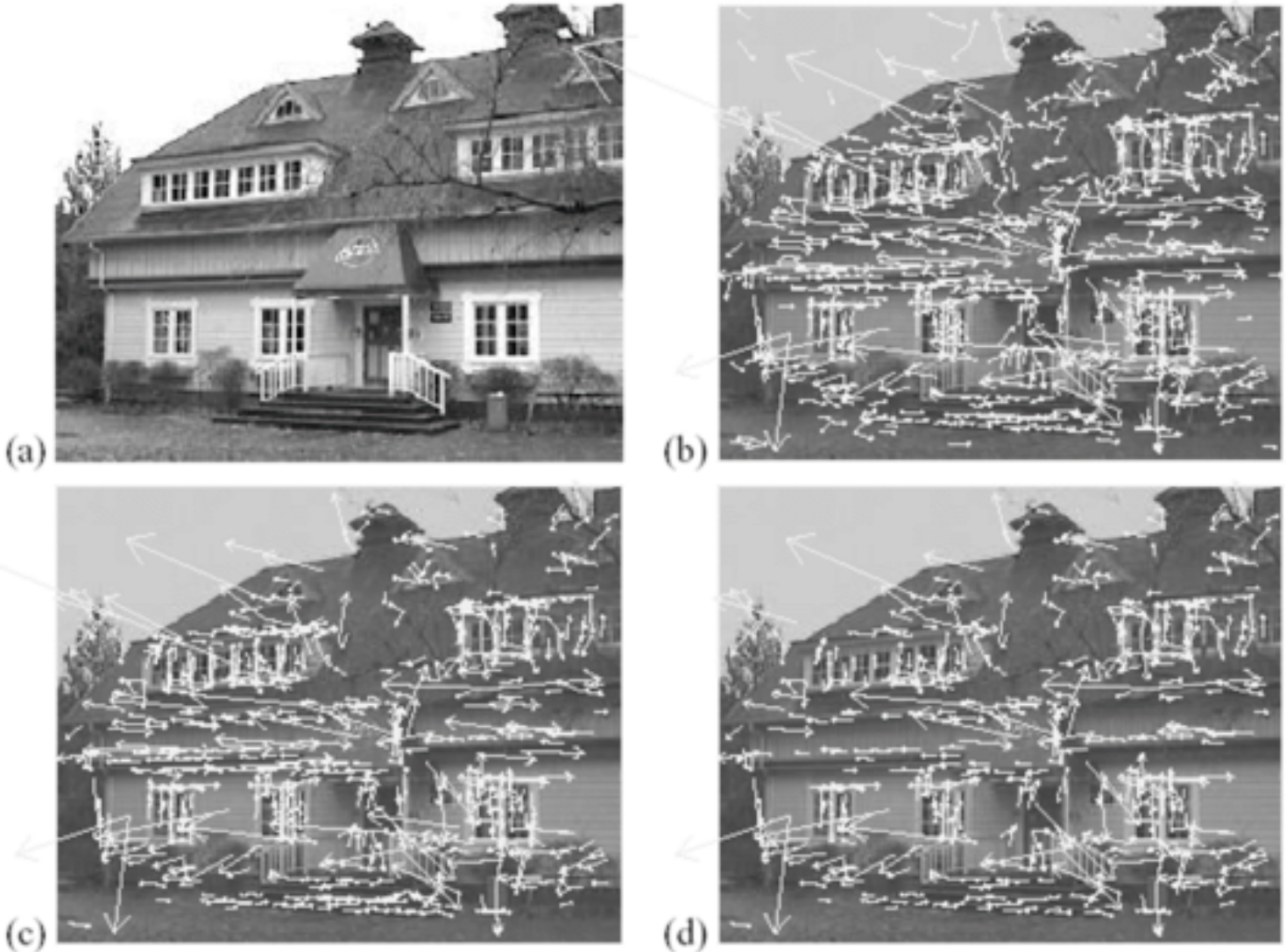
(Scale-Invariant Feature Transform, con't)

The goal is to estimate a homography: to find the rotation, translation, and scale parameters that best relate the contents of two image frames.

Key idea: identifying keypoints that correspond in different images, and discovering transformations that map them to each other.

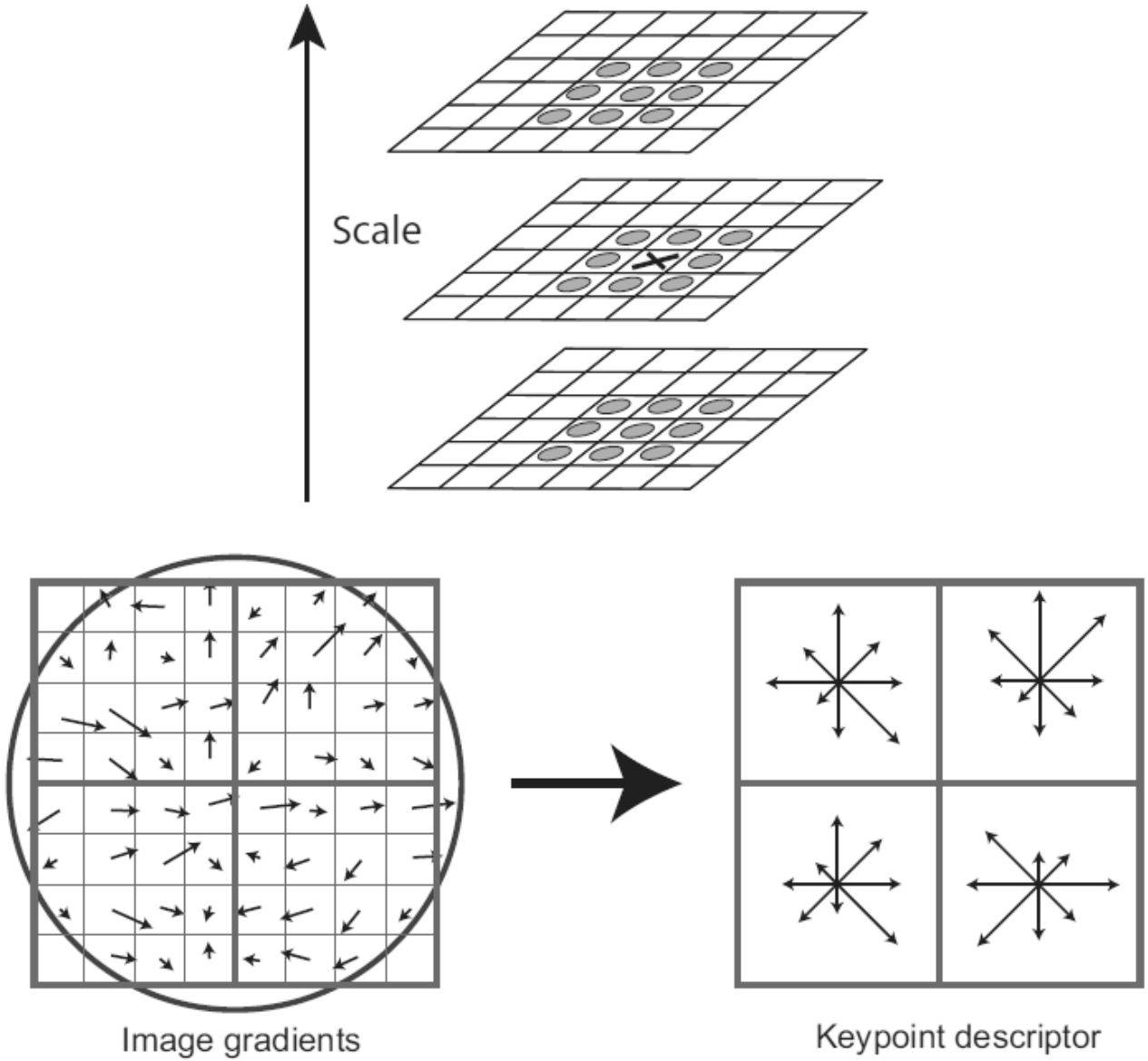
- Various kinds of feature detectors can be used, but they should have an orientation index and a scale index
- Classic approach of Lowe used extrema (maxima and minima) of difference-of-Gaussian functions in scale space
- Build a Gaussian image pyramid in scale space by successively smoothing (at octave blurring scales $\sigma_i = \sigma_0 2^i$) and resampling
- Dominant orientations of features, at various scales, are detected and indexed by oriented edge detectors (*e.g.* gradient direction)
- Low contrast candidate points and edges are discarded
- The most stable keypoints are kept, indexed, and stored for "learning" a library of objects or classes

(Scale-Invariant Feature Transform, con't)



Examples of keypoints (difference-of-Gaussian extrema) detected in an original image, of which 35% are discarded as low contrast or unstable.

(Scale-Invariant Feature Transform, con't)



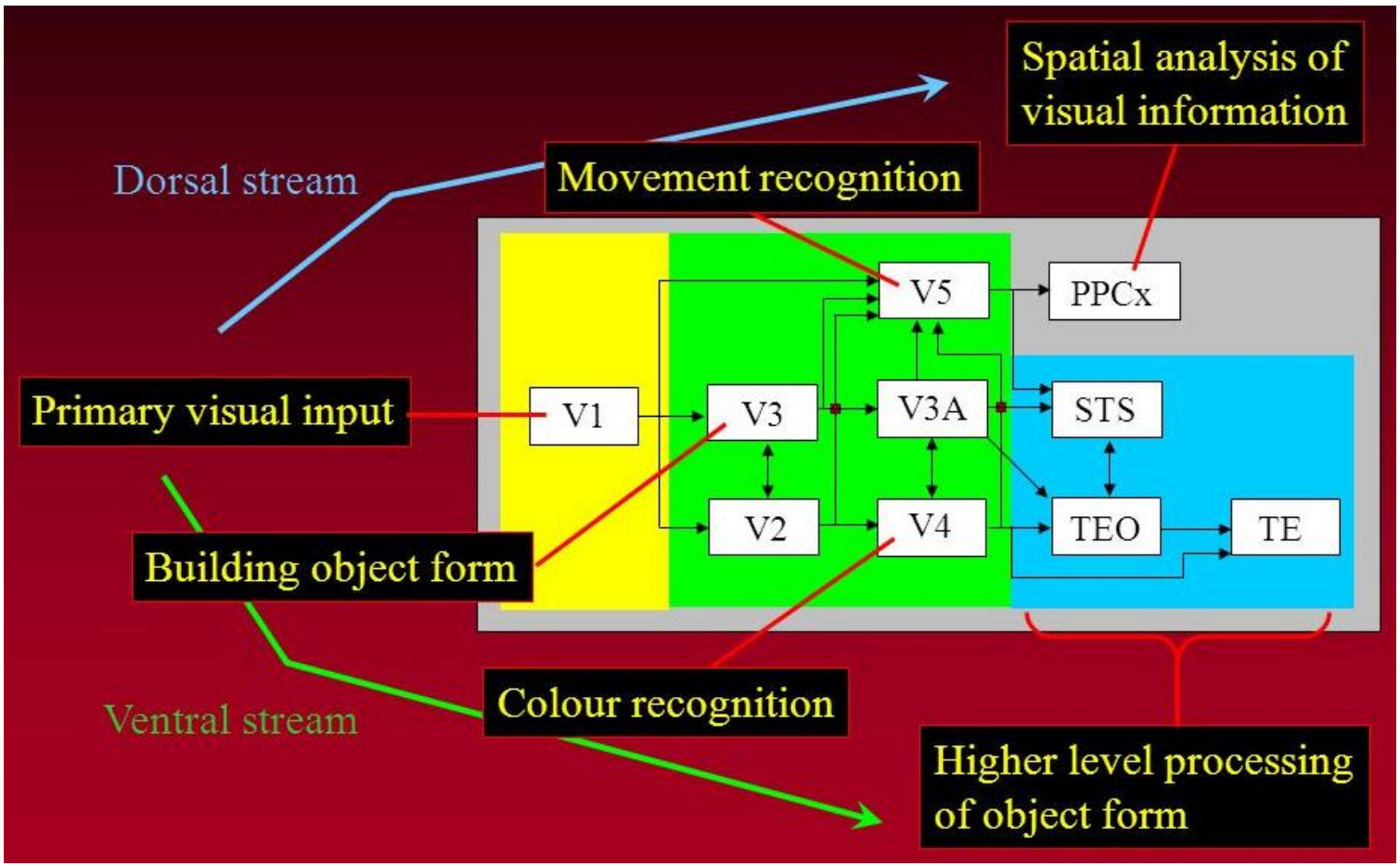
For each local region (four are highlighted here), an orientation histogram is constructed from the gradient directions as a keypoint descriptor.

(Scale-Invariant Feature Transform, con't)

- The bins of the orientation histogram are normalised relative to the dominant gradient direction in the region of each keypoint, so that rotation-invariance is achieved
- Matching process resembles identification of fingerprints: compare relative configurations of groups of minutiae (ridge terminations, spurs, etc), but search across many relative scales as well
- The best candidate match for each keypoint is determined as its nearest neighbour in a database of extracted keypoints, using the Euclidean distance metric
- Algorithm: best-bin-first; heap-based priority queue for search order
- The probability of a match is computed as the ratio of that nearest neighbour distance, to the second nearest (required ratio > 0.8)
- Searching for keys that agree on a particular model pose is based on Hough Transform voting, to find clusters of features that vote for a consistent pose
- SIFT does not account for any non-rigid deformations
- Matches are sought across a wide range of scales and positions; 30 degree orientation bin sizes; octave (factor of 2) changes in scale

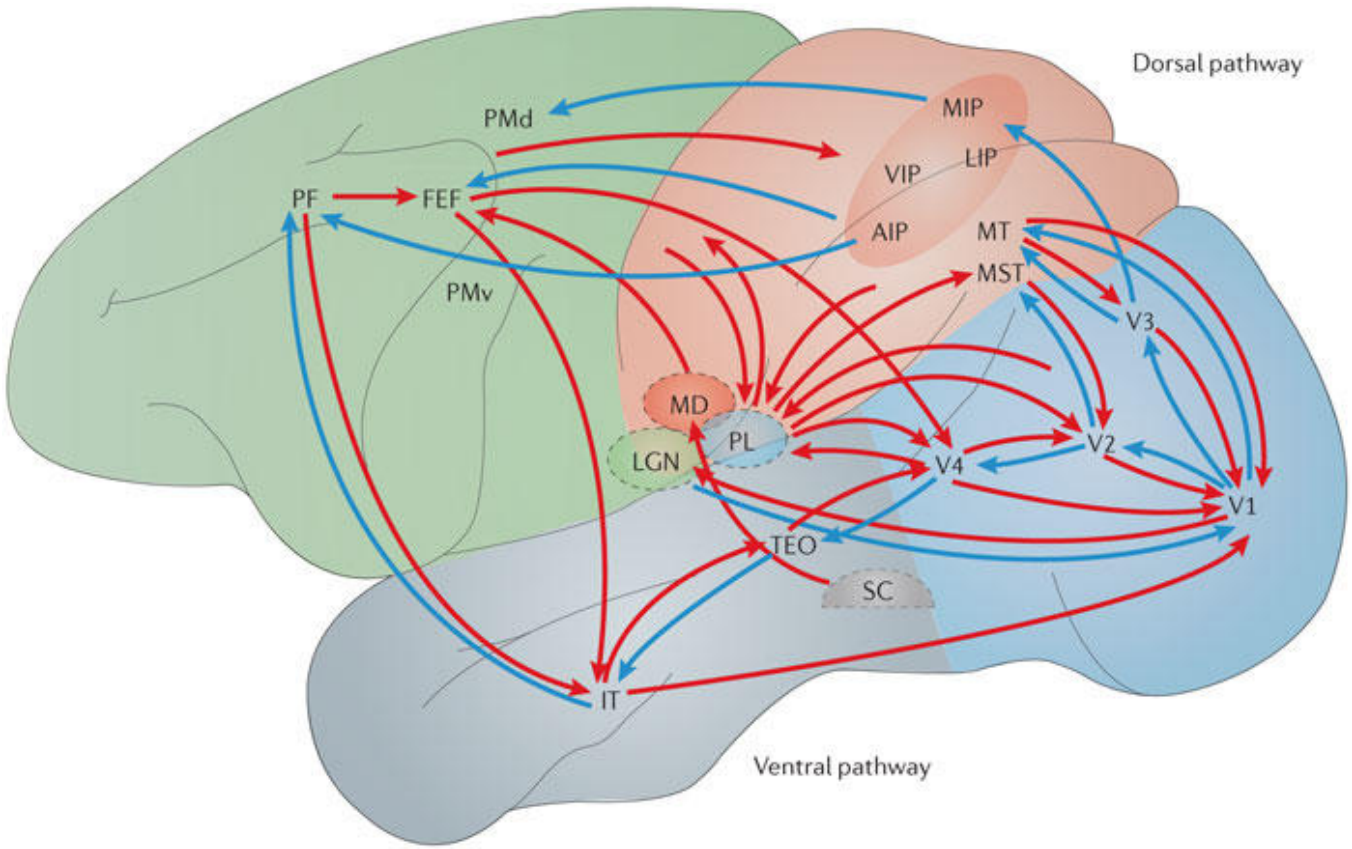
7. Parallel functional streams; reciprocal feedback

- Parallel functional streams exist in the brain, specialised for visual subdomains such as form, colour, motion, and texture processing.
- There exist 30+ distinct, specialised visual brain areas. These can be grouped broadly into "dorsal" and "ventral" hierarchies. A further dichotomy may exist between "conscious" and "unconscious" vision.



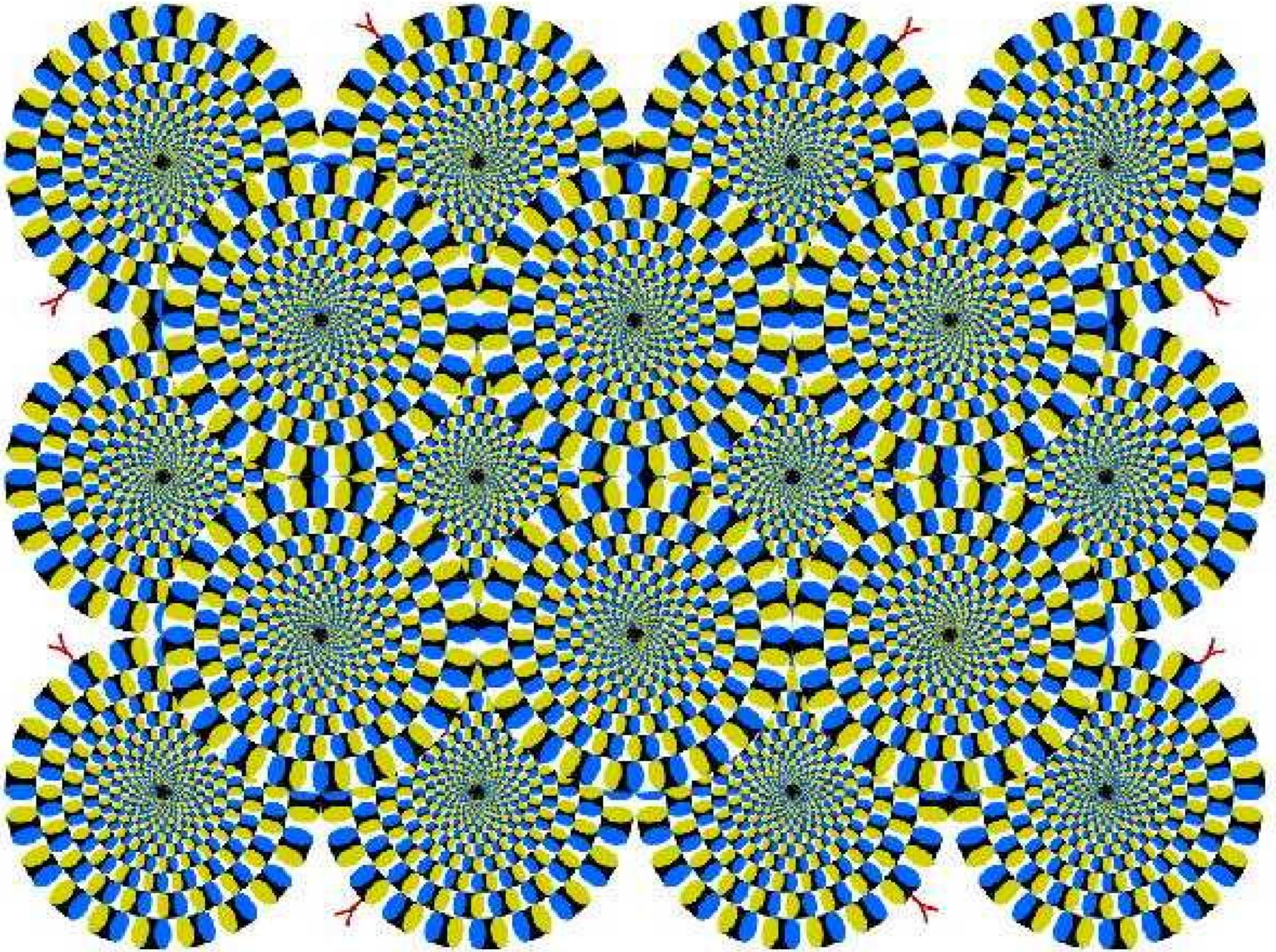
(7. Parallel functional streams; reciprocal feedback, con't)

- It is not unreasonable to ask how results from the division-of-labour get reunified later. (E.g. analysing a bird in flight: how do its form, colour, texture, and motion information get "put back together"?)
- Reciprocal pairwise connections exist between many such areas, highlighted here by blue and red pairings of arrows.



See this 3D scan reconstruction of wiring bundles connecting diverse parts of the human brain:
www.bbc.co.uk/news/video_and_audio/must_see/40487049/the-most-detailed-scan-of-the-wiring-of-the-human-brain

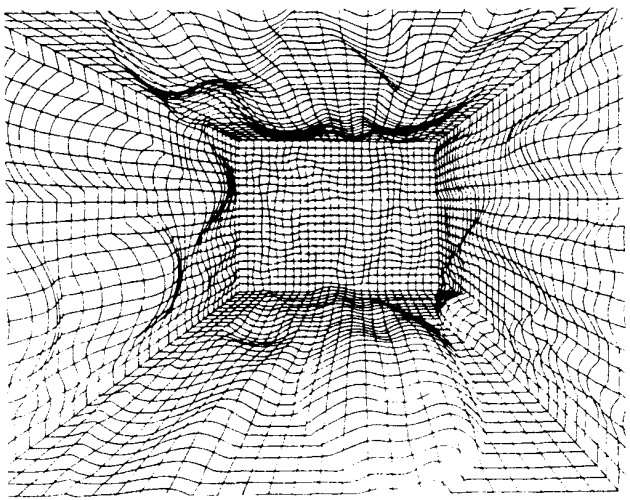
Interactions between colour, motion, & texture streams?

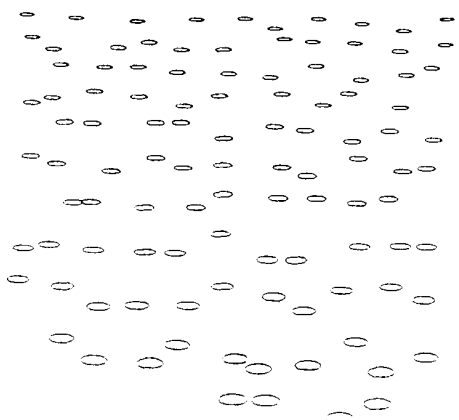


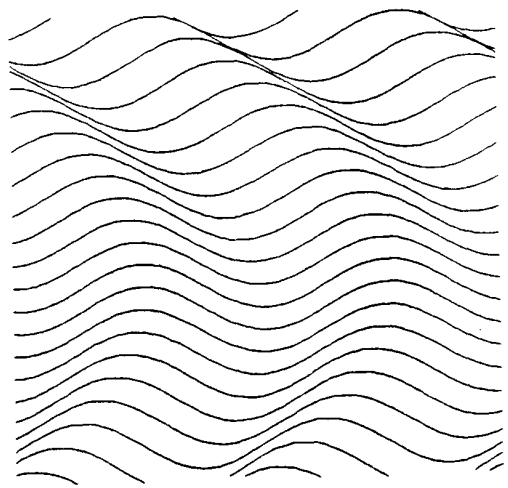
It even works for cats: `https://youtu.be/S4IHB3qK1KU`

8a. Structure from texture

- Most surfaces are covered with texture, of one sort or another
- Texture is both an identifying feature, and a cue to surface shape
- If one can assume uniform statistics along the surface itself, then textural foreshortening or stretching reveals 3D surface shape
- As implied by its root, linking it with (woven) textiles, texture is defined by the existence of statistical correlations across the image
- From grasslands to textiles, the unifying notion is quasi-periodicity
- Variations from uniform periodicity reveal 3D shape, slant, distance



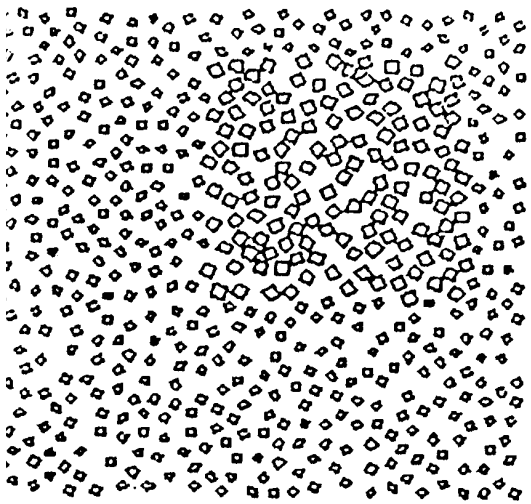
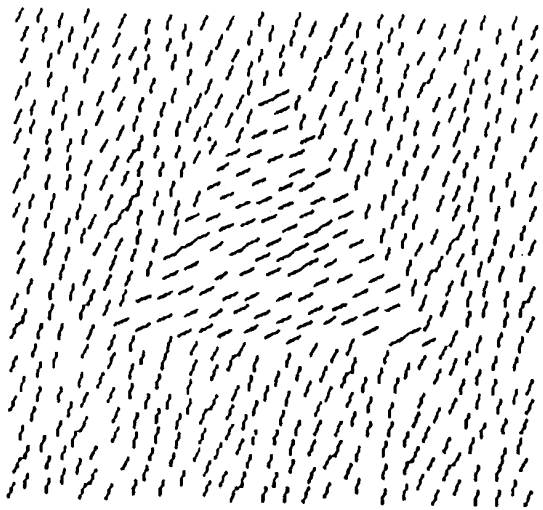
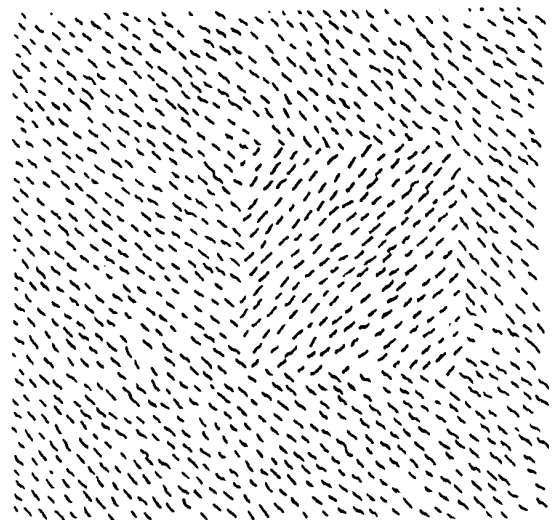




(Structure from texture, con't)

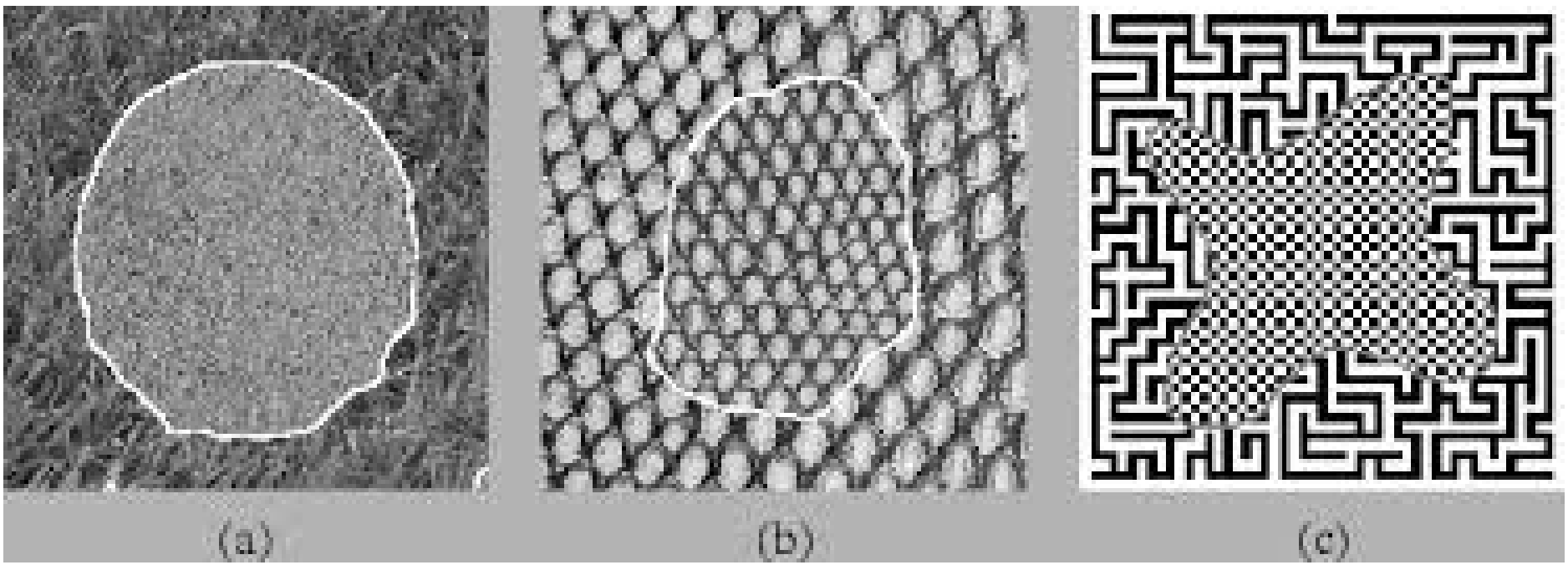
- Quasi-periodicity can be detected best by Fourier-related methods
- The eigenfunctions of Fourier analysis (complex exponentials) are periodic, with a specific scale (frequency) and wavefront orientation
- Therefore they excel at detecting a correlation distance and direction
- They can estimate the "energy" within various quasi-periodicities

- Texture also supports figure/ground segmentation by dipole statistics
- The examples below can be segmented (into figure vs ground) either by their first-order statistics (size of the texture elements), or by their second-order statistics (dipole orientation)

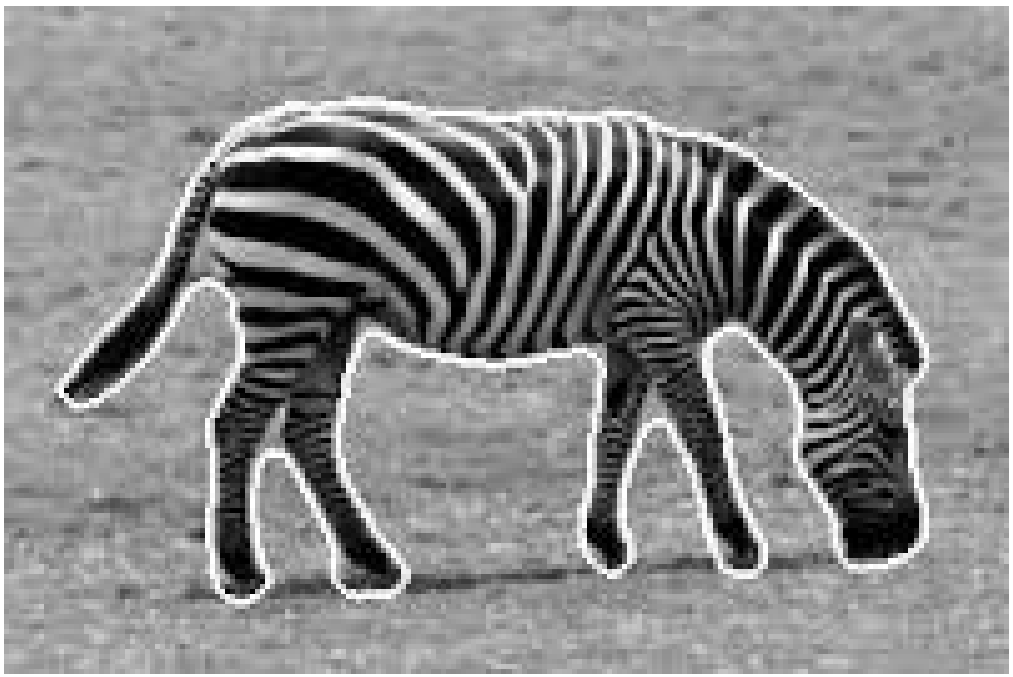
(Structure from texture, con't)

- Images can be segmented into "figure" vs "ground" regions using Gabor wavelets of varying frequencies and orientations
- The modulus of Gabor wavelet coefficients reveals texture energy variation in those frequencies and orientations across the image
- This can be a strong basis for image segmentation (outlined regions)



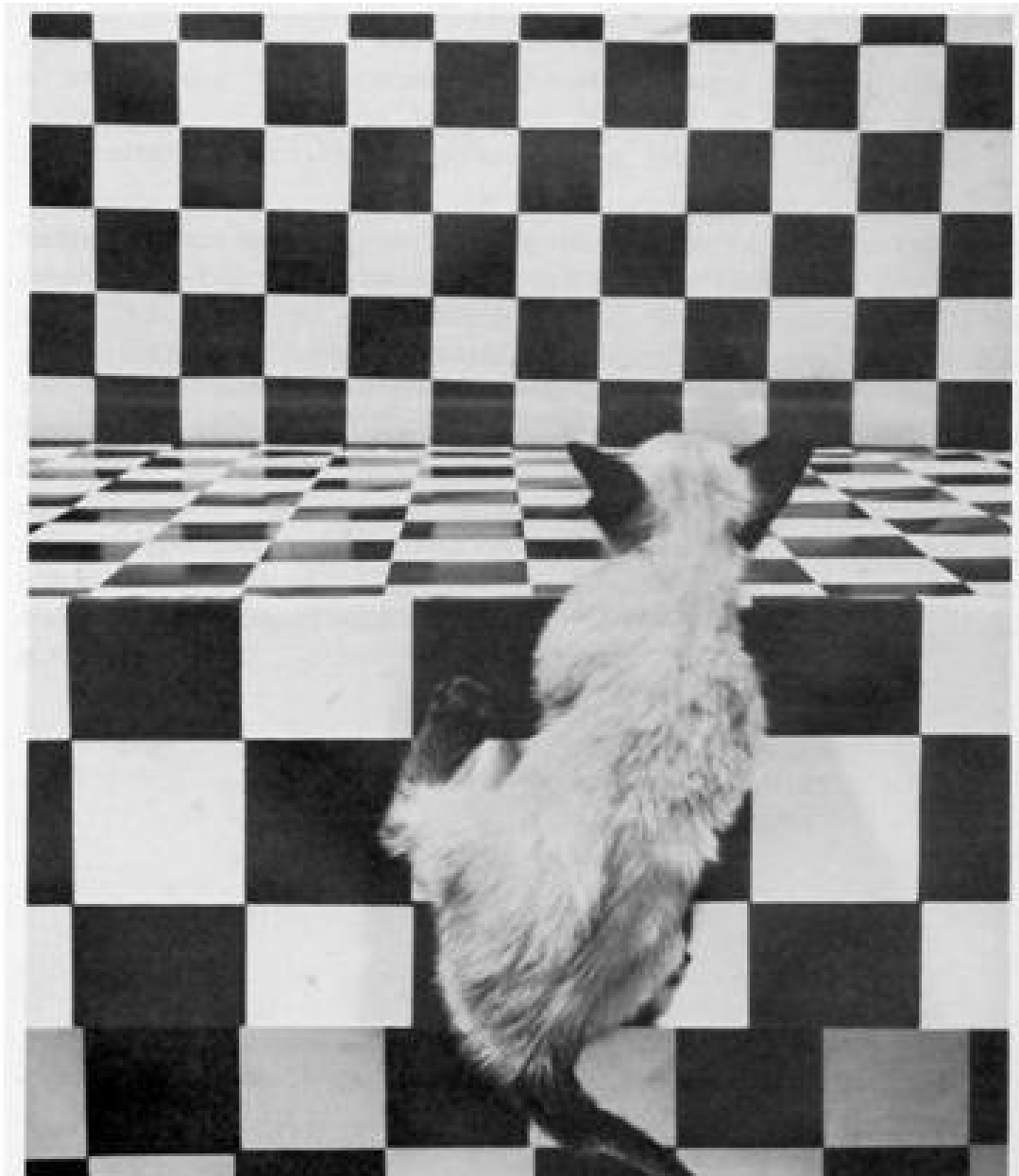
(Structure from texture, con't)

- Resolving textural spectra simultaneously with location information is limited by the Heisenberg Uncertainty Principle, and this trade-off is optimised by Gabor wavelets
- Texture segmentation using Gabor wavelets can be a basis for extracting the shape of an object to recognise it. (Left image)
- Phase analysis of iris texture using Gabor wavelets is a powerful basis for person identification. (Right image)



(Structure from texture, con't)

Inferring depth from texture gradients can have real survival value...



8b. Colour information

Two compelling paradoxes are apparent in how humans process colour:

1. Perceived colours hardly depend on the wavelengths of illumination (colour constancy), even with dramatic changes in the wavelengths
2. But the perceived colours depend greatly on the local context

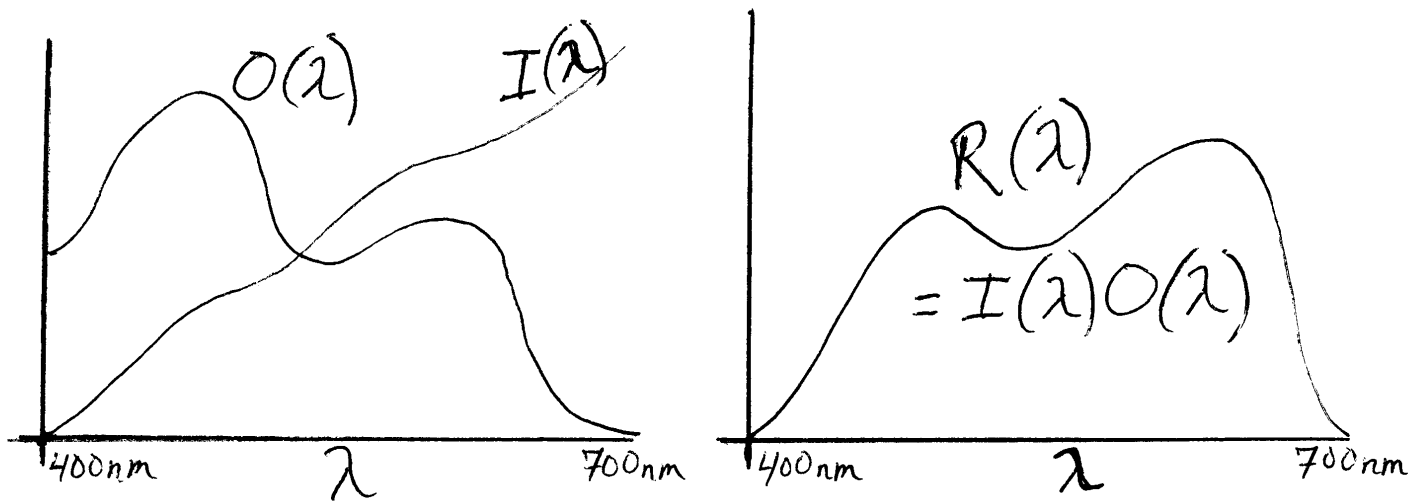
The brown tile at the centre of the illuminated upper face of the cube, and the orange tile at the centre of the shadowed front face, are actually returning the same light to the eye (as is the tan tile lying in front)

(Colour information, con't)

Colour is a nearly ubiquitous property of surfaces, and it is useful both for object identification and for segmentation. But inferring colour properties ("spectral reflectances") of object surfaces from images seems impossible, because generally we don't know the spectrum of the illuminant.

- Let $I(\lambda)$ be the wavelength composition of the illuminant
- Let $O(\lambda)$ be the spectral reflectance of the object at some point (the fraction of light scattered back as a function of wavelength λ)
- Let $R(\lambda)$ be the actual wavelength mixture received by the camera at the corresponding point in the image, say for $(400\text{nm} < \lambda < 700\text{nm})$

Clearly, $R(\lambda) = I(\lambda)O(\lambda)$. The problem is that we wish to infer the "object colour" $O(\lambda)$, but we only know $R(\lambda)$, the mixture received.



(Colour information, con't)

An algorithm for computing $O(\lambda)$ from $R(\lambda)$ was proposed by Dr E Land (founder of Polaroid Corporation). He named it the *Retinex Algorithm* because he regarded it as based on biological vision (RETINa + cortEX).

It is a ratiometric algorithm:

1. Obtain the red/green/blue value (r, g, b) of each pixel in the image
2. Find the maximal values $(r_{max},\ g_{max},\ b_{max})$ across all the pixels
3. Assume that the scene contains some objects that reflect "all" the red light, others that reflect "all" the green, and others "all" the blue
4. Assume that those are the origins of the values $(r_{max},\ g_{max},\ b_{max})$, thereby providing an estimate of $I(\lambda)$
5. For each pixel, the measured values (r, g, b) are assumed to arise from actual object spectral reflectance $(r/r_{max},\ g/g_{max},\ b/b_{max})$
6. With this renormalisation, we have discounted the illuminant
7. Alternative variants of the Retinex exist which estimate $O(\lambda)$ using only local comparisons across colour boundaries, assuming only local constancy of the illuminant spectral composition $I(\lambda)$, rather than relying on a global detection of $(r_{max},\ g_{max},\ b_{max})$

(Colour information, con't)

Colour assignments are very much a matter of calibration, and of making assumptions. Many aspects of colour are "mental fictions".

For example, why does perceptual colour space have a seamless, cyclic topology (the "colour wheel"), with red fading into violet fading into blue, when in wavelength terms that is moving in *opposite* directions along a line ($\lambda \rightarrow$ 700nm red) versus (blue 400nm $\leftarrow \lambda$)?

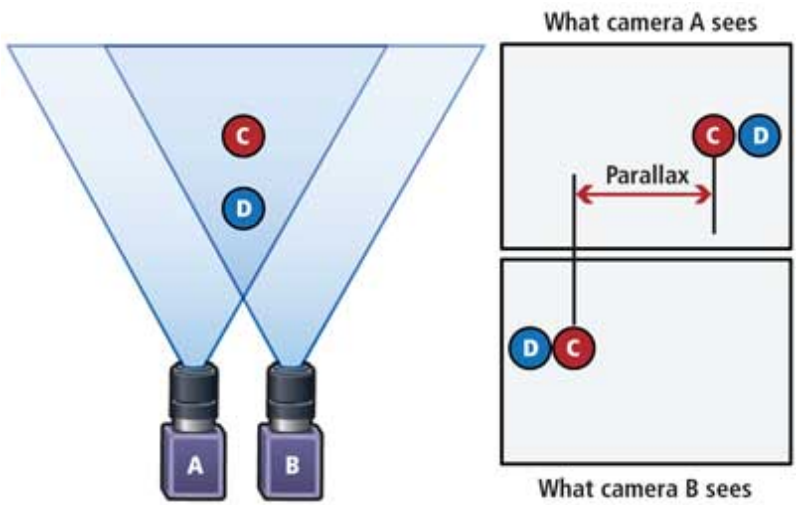
The next slide is a purely monochromatic (black-and-white) picture. But you can cause it to explode into compelling colours by re-calibrating your brain, using the *subsequent* false colour image (2 slides ahead):

1. Stare at the blue disk in the false colour image for about 10 seconds, without moving your eyes. (Finger on key, ready to "flip back")
2. Flip back to the monochromatic image, while continuing to fixate on that same central point
3. As long as you don't move your eyes, you should see very rich and compelling and appropriate colours in the monochromatic image
4. The spell will be broken, your brain's original calibration restored, once you move your eyes

8c. Structure from stereo vision

An important source of information about the 3D structure of the surrounding (near) visual world is stereo vision, using stereo algorithms

- Having 2 (or more) cameras, or 2 eyes, with a base of separation, allows the capture of simultaneous images from different positions
- Such images have differences called stereoscopic disparity, which depend on the 3D geometry of the scene, and on camera properties
- 3D depth information can be inferred by detecting those differences, which requires solving the correspondence problem



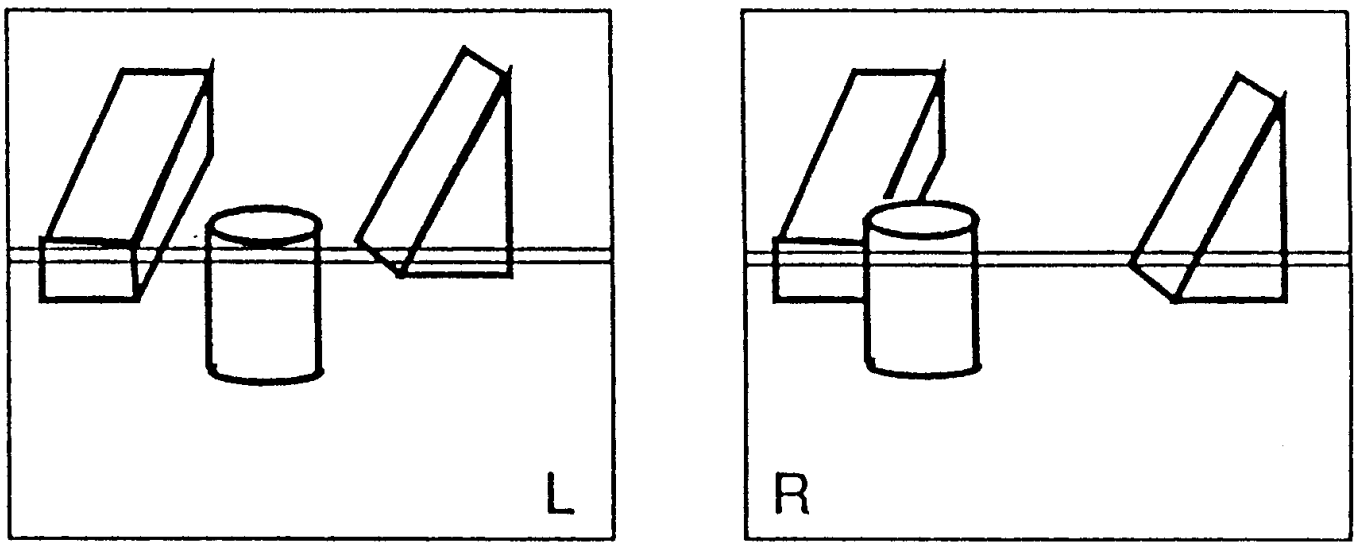
(Structure from stereo vision, con't)

Of course, alternative methods exist for estimating depth. For example, the *"Kinect"* gaming device projects an infrared (IR, invisible) laser grid into the scene, whose resulting pitch in the image sensed by an IR camera is a cue to depth and shape, as we saw in discussing shape from texture. Here we consider only depth computation from stereoscopic disparity.

- Solving the correspondence problem can require very large searches for matching features under a large number of possible permutations
- We seek a relative registration which generates maximum correlation between the two scenes acquired with the spatial offset, so that their disparities can then be detected and measured
- A multi-scale image pyramid (several resolutions) is helpful here
- It steers the search by a coarse-to-fine strategy to maximise its efficiency, as only few features are needed for a coarse-scale match
- The permutation-matching space of possible corresponding points is greatly attenuated, before refining the matches iteratively, ultimately terminating with single-pixel precision matches

(Structure from stereo vision, con't)

- If the optical axes of the 2 cameras converge at a point, then objects in front or behind that point in space will project onto different parts of the two images. This is sometimes called parallax
- The disparity becomes greater in proportion to the distance of the object in front, or behind, the point of fixation
- Clearly it depends also on the convergence angle of the optical axes
- Even if the optical axes parallel each other ("converged at infinity"), there will be disparity in the image projections of nearby objects
- Disparity also becomes greater with increased spacing between the two cameras, as that is the base of triangulation

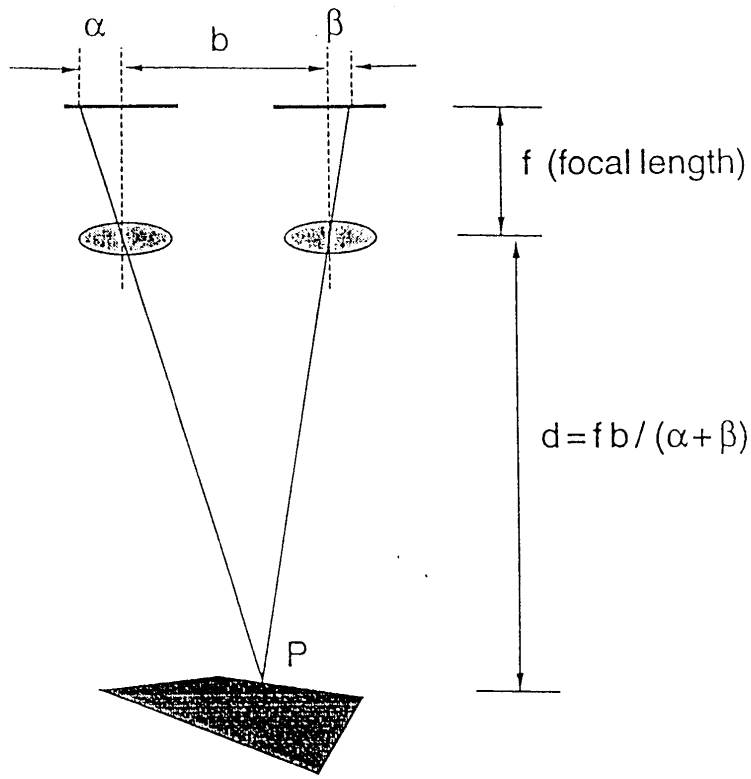


(Structure from stereo vision, con't)

In the simplifying case that the optical axes are parallel, once the correspondence problem has been solved, plane geometry enables calculation of how the depth d of any given point depends on:

- camera focal length f
- base distance b between the optical centres of their lenses
- disparities (α, β) in the image projections of some object point (P) in opposite directions relative to the optical axes, outwards

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



Note: P is "at infinity" if $(\alpha, \beta) = 0$

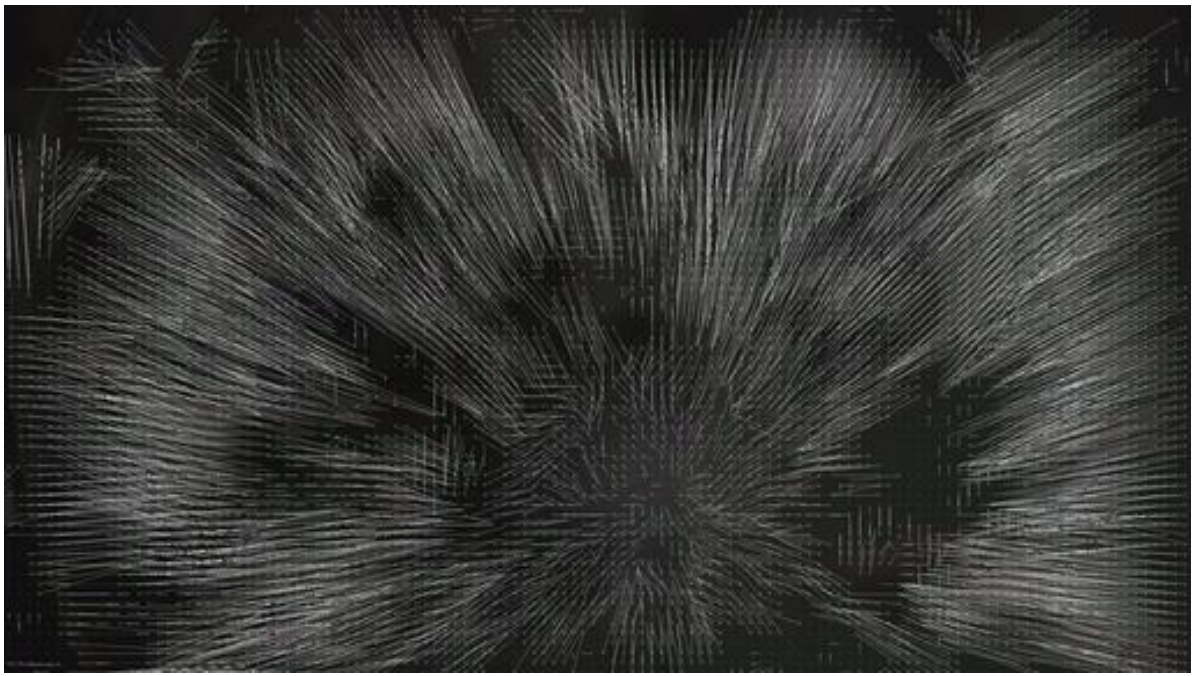
(Structure from stereo vision, con't)

In World War I, trench stereo periscopes were used not only to peer "safely" over the parapets, but by increasing the base of triangulation (increasing the angle of the V-shape), to try to "break camouflage".

Imagine the visual world of "hunter spiders" that have got eight eyes...

8d. Optical flow; detecting and estimating motion

- Optical flow is the apparent motion within a visual scene that is caused by relative motion between an observer and the scene.
- It assists scene understanding, segmentation, 3D object recognition, stereo vision, navigation control, collision and obstacle avoidance.
- In general we have both ego-motion of the viewer (camera), and also independent motion of objects in the scene, with combined effects.
- Motion estimation computes local motion vectors that describe the transformation between frames in a video sequence. It is a variant of the correspondence problem, illustrated by this vector field:



Information from motion vision

Few vision applications involve just static image frames. That is basically vision "off-line;" – but the essence of an effective visual capability is its real-time use in a dynamic environment. Among the challenges are:

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

  Video demo of motion-based tracking: https://www.youtube.com/watch?v=oL67qe-Fhps
- It may be necessary to assign more than one velocity vector to any given local image region (as occurs in "motion transparency").
- Need to disambiguate object motion from contour motion, so that we can measure the velocity of an object regardless of its form.
- We may need to detect a coherent overall motion pattern across many small objects or regions separated from each other in space.
- May need complex inferences about form and object identity, from merely a few moving points. See classic Johansson demonstration of easily identifiable human activity from just a few sparse points: http://www.youtube.com/watch?v=r0kLC-pridI - even gender and age: https://www.youtube.com/watch?v=4E3JdQcmIAg (- is he a Neanderthal?)

Main classes of motion detection and estimation models

- Intensity gradient models: Assume that the local time-derivative in image intensities at a point is related to the local spatial gradient in $I(x, y, t)$ image intensities because of some 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 $\vec{\nabla} I(x, y, t)$ is an estimate of the local image velocity (in the direction of the gradient).

- Dynamic zero-crossing models: Measure image velocity by finding first 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 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 spectral methods for motion estimation

- Motion can also be detected and measured by Fourier methods.
- 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 3-dimensional (spatio-temporal) Fourier transform is defined:

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

- Rigid motion in an area 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 is detected by applying spatio-temporal filters to an image sequence and observing that filters whose centre frequencies are co-planar in this 3-space are activated together.
- The azimuth and elevation of that inclined spectral plane correspond respectively to the direction and speed of the motion.

Spectral co-planarity theorem

Translational image motion has a 3D spatio-temporal Fourier spectrum that is non-zero only on a plane through the origin of frequency-space. Coordinates of the unit normal to this spectral plane correspond to the speed and direction of motion.

1. Let $I(x, y, t)$ be a continuous image in space and time.
   Let $F(\omega_x, \omega_y, \omega_t)$ be its 3D spatio-temporal Fourier transform:

$$F(\omega_x, \omega_y, \omega_t) = \int_x \int_y \int_t I(x, y, t) e^{-i(\omega_x x + \omega_y y + \omega_t t)} dt\, dy\, dx.$$

   We wish to detect local image velocity $\vec{\mathbf{v}} = (v_x, v_y)$ in $I(x, y, t)$.
2. Uniform motion $\vec{\mathbf{v}}$ implies space shifts with time shifts t_o:

$$I(x, y, t) = I(x - v_x t_o, y - v_y t_o, t - t_o).$$

3. Taking the 3D spatio-temporal Fourier transform of both sides, and applying the shift theorem, gives

$$F(\omega_x, \omega_y, \omega_t) = e^{-i(\omega_x v_x t_o + \omega_y v_y t_o + \omega_t t_o)} F(\omega_x, \omega_y, \omega_t).$$

(Spectral co-planarity theorem, con't)

4. The last equation above can only be true if $F(\omega_x, \omega_y, \omega_t) = 0$ everywhere the exponential factor doesn't equal 1.
5. This means $F(\omega_x, \omega_y, \omega_t)$ is non-zero only on the 3D spectral plane $\boxed{\omega_x v_x + \omega_y v_y + \omega_t = 0}$ (or on parallel copies of it separated by 2π).
6. The elevation, or slope of this plane relative to the spatial frequency plane (ω_x, ω_y), corresponds to the speed of motion:

$$\text{speed} = \sqrt{v_x^2 + v_y^2}$$

7. The azimuth, or direction of tilt of this plane projected as a ray into the spatial frequency plane (ω_x, ω_y), corresponds to the direction of the motion:

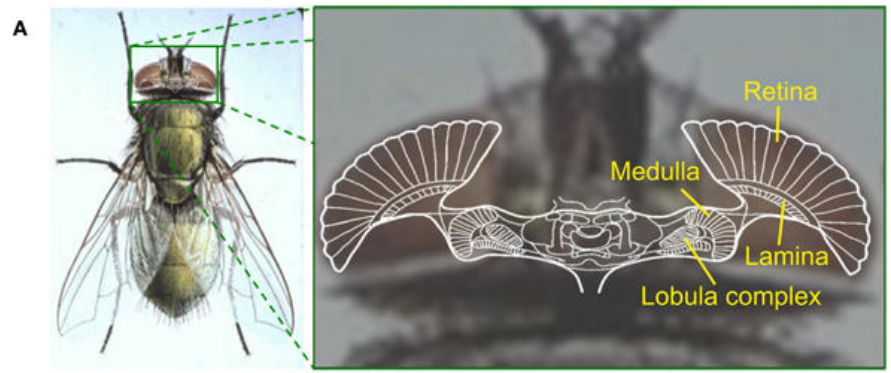
$$\text{direction} = \tan^{-1}(v_y / v_x)$$

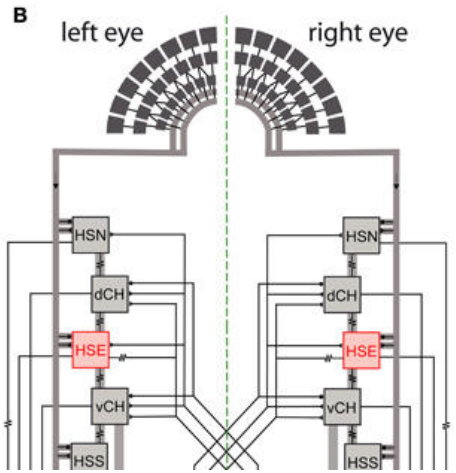
□

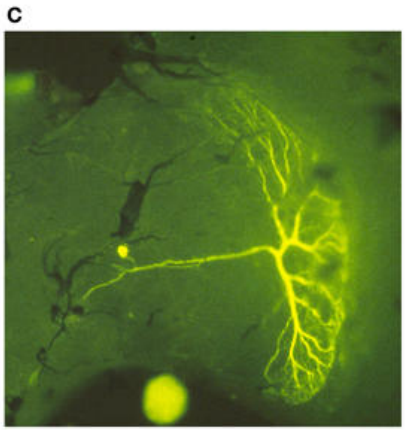
Thus we see that observing a geometrical relationship (i.e. co-planarity) among the spatio-temporal filter frequencies that are energetic during an image sequence, reveals motion, and estimates its speed and direction.

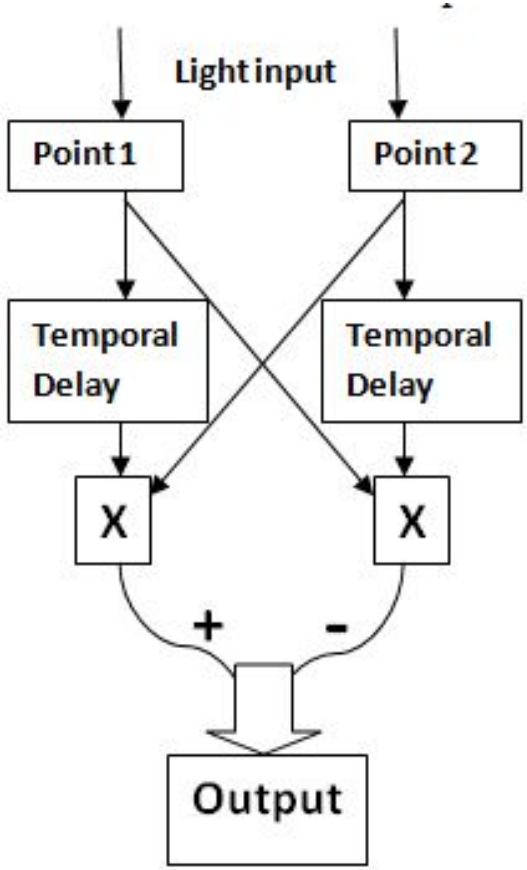
Spatio-temporal correlation models for motion detection

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

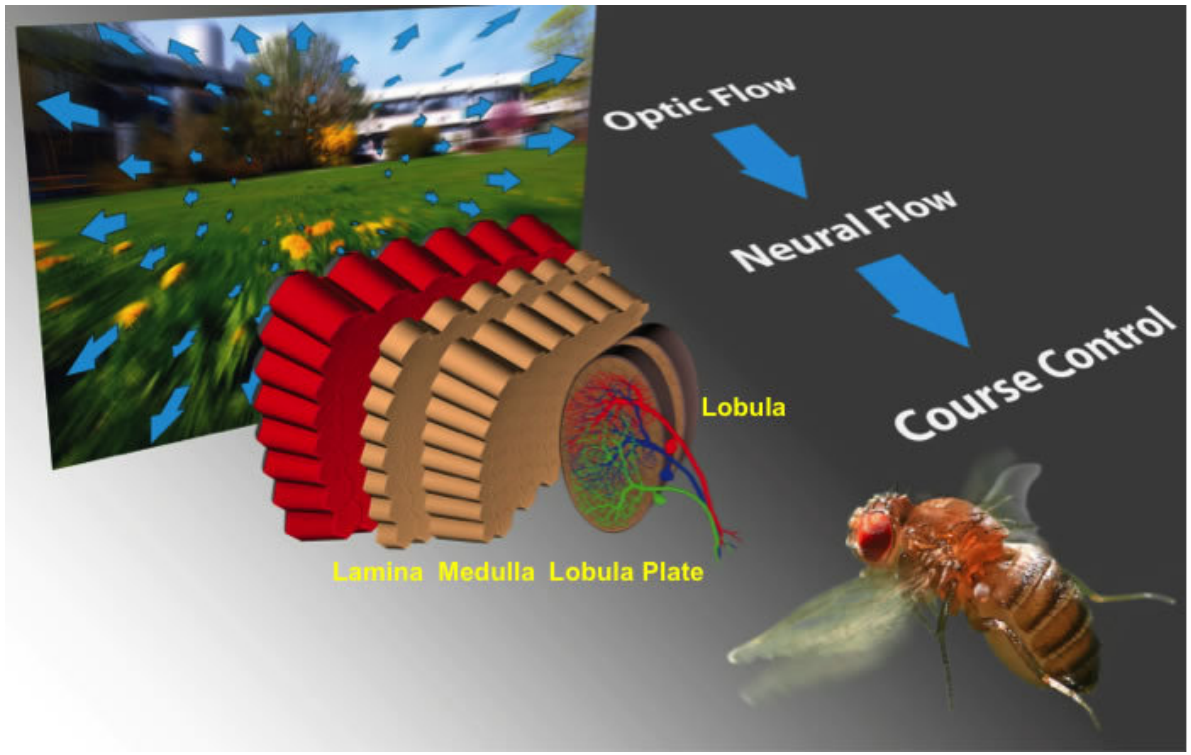








Optical flow: elementary motion detectors in flying insects (and for other autonomous vehicles like driverless cars?)



Two approaches for autonomous vehicles to map the local topology –

- SLAM (Simultaneous Localisation And Mapping): probabilistically updated maps, often using stereo vision and Monte Carlo methods.
- LIDAR (LIght Detection And Ranging): emits scanning laser pulses, measures time of detected reflection from objects ($c \approx 1$ foot/nsec).

9. Surfaces and reflectance maps

How can we infer the shape and reflectance properties of a surface from measurements of brightness in an image?



This is complicated because many factors besides shape determine how (and where) objects scatter light.

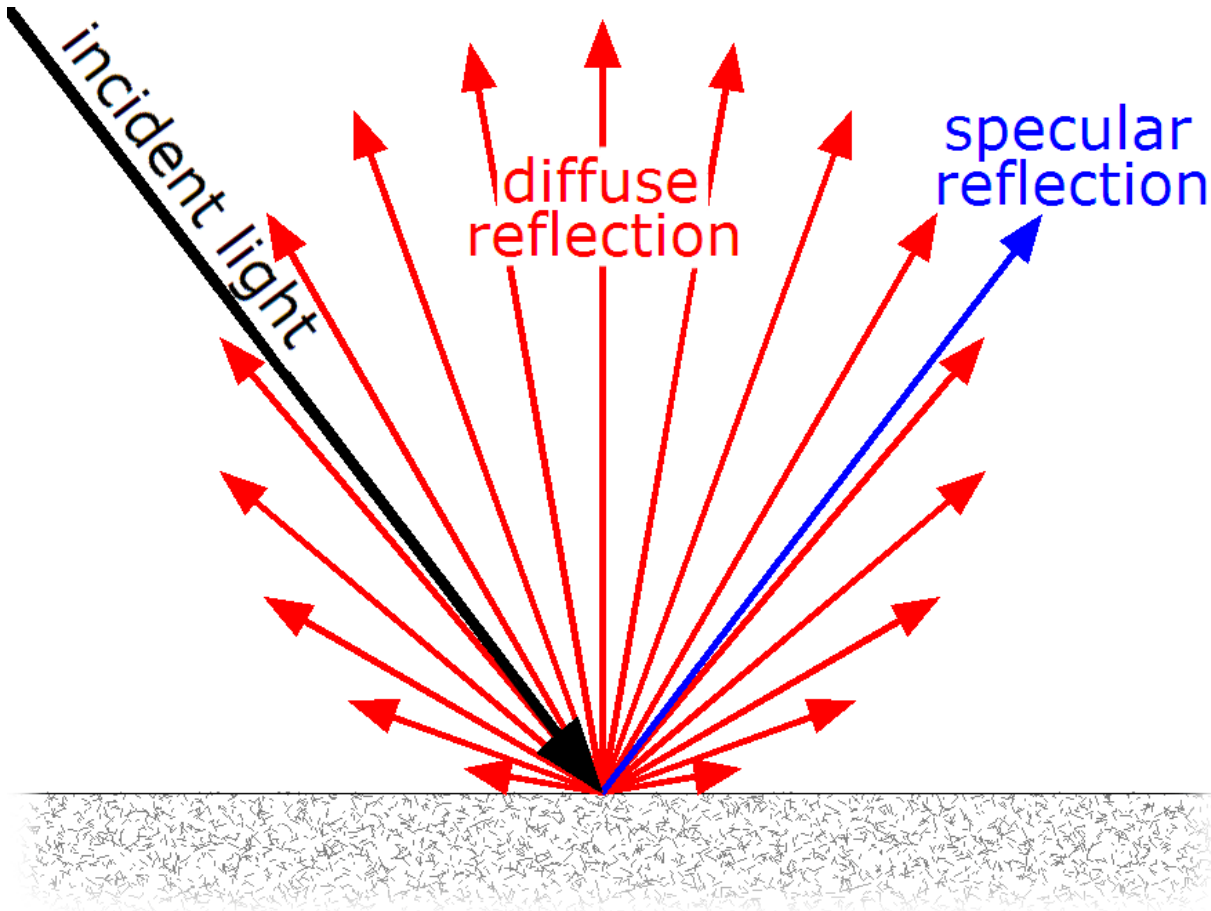
- Surface albedo is the fraction of the illuminant that is re-emitted from a surface in all directions. Thus it relates to how "light" or "dark" is the surface, and this may vary locally across it
- The amount of light reflected is the product of two factors: the surface albedo, times a geometric factor that depends on angles

(Surfaces and reflectance maps, con't)

- A Lambertian surface (also called diffusely reflecting, or "matte") reflects light equally well in all directions
- Examples of Lambertian surfaces include: snow, non-glossy paper, ping-pong balls, magnesium oxide, projection screens, ...
- The amount of light reflected from a Lambertian surface depends on the angle of incidence of the light (by Lambert's famous cosine law), but not on the angle of emission (the viewing angle)
- A specular surface is mirror-like. It obeys Snell's law (the angle of incidence of light is equal to its angle of reflection from the surface), and it does not scatter light into other angles
- Many metallic surfaces are specular. But more generally, surfaces lie somewhere on a continuum between Lambertian and specular
- Special cases arise from certain kinds of dust. The surface of the moon (called unsurprisingly a lunar surface) reflects light depending on the ratio of cosines of angle of incidence and angle of emission
- That is why a full moon looks more like a penny than like a sphere; its brightness does not fade, approaching the boundary (!)

(Surfaces and reflectance maps, con't)

Geometric summary of Lambertian, versus specular, properties of surfaces

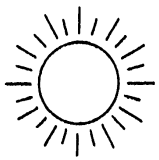


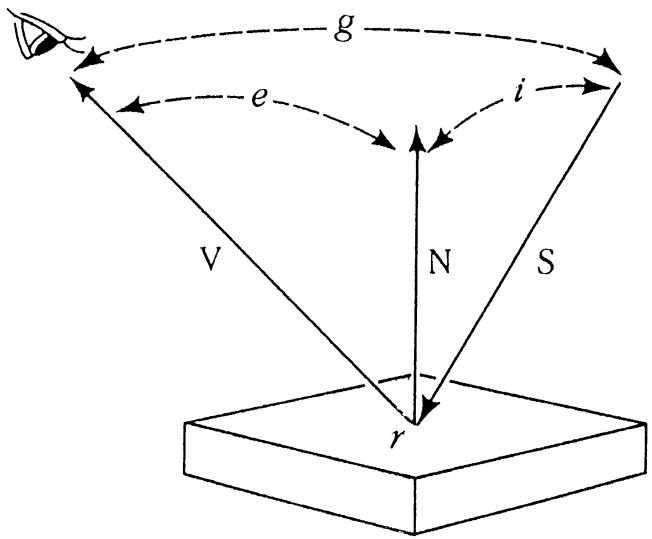
(Surfaces and reflectance maps, con't)

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:

- 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
- g is the angle between the emitted ray and the illuminant





(Surfaces and reflectance maps, con't)

There are many types of reflectance maps $\phi(i, e, g)$, each of which is characteristic of certain surfaces and imaging environments

- Lambertian surface: reflectance function is $\phi(i, e, g) = \cos(i)$
  (It looks equally bright viewed from all directions; the amount of reflected light depends only on the angle of illumination)
- Specular surface: $\phi(i, e, g)$ 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 perfect mirror); otherwise $\phi(i, e, g) = 0$
- For "lunar" surfaces such as the feldspar dusts on 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 wondered, this is why the full moon looks like a penny rather than a sphere. Even though it is illuminated by a point source (the sun, behind you), it does not fade in brightness approaching its limb (boundary) as the surface normal N tilts, because still $i = -e$

(Surfaces and reflectance maps, con't)

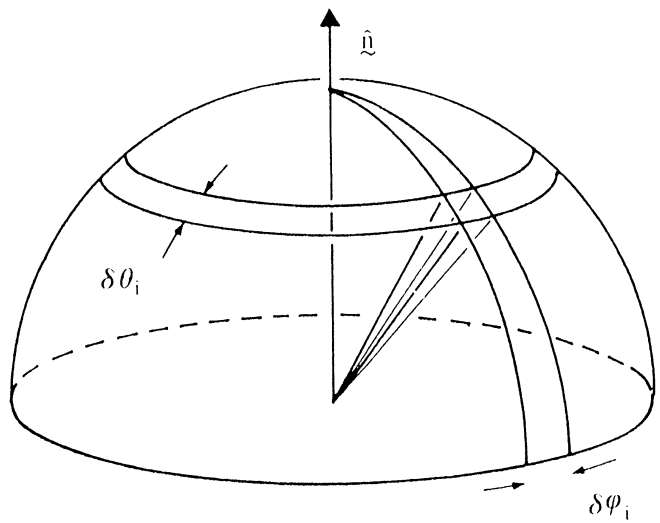
Typically, surfaces have both specular and matte properties. For example, facial skin may vary from Lambertian (powdered) to specular (oily). The purpose of powdering one's face is to specify s and n in this expression:

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

- Linear combination of two terms, with weights s and $(1-s)$
- The first term on the right side is the specular component
- The second term on the right side is the Lambertian component
- s is the fraction of light emitted specularly
- n represents the sharpness (in angle) of the specular peak
- For glossy paint, typically the exponent n may be about 20
- Obviously as n grows very large, the exponentiated trigonometric function approaches a delta function, representing Snell's law for mirrors: a very sharp power function of angle

(Surfaces and reflectance maps, con't)

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.



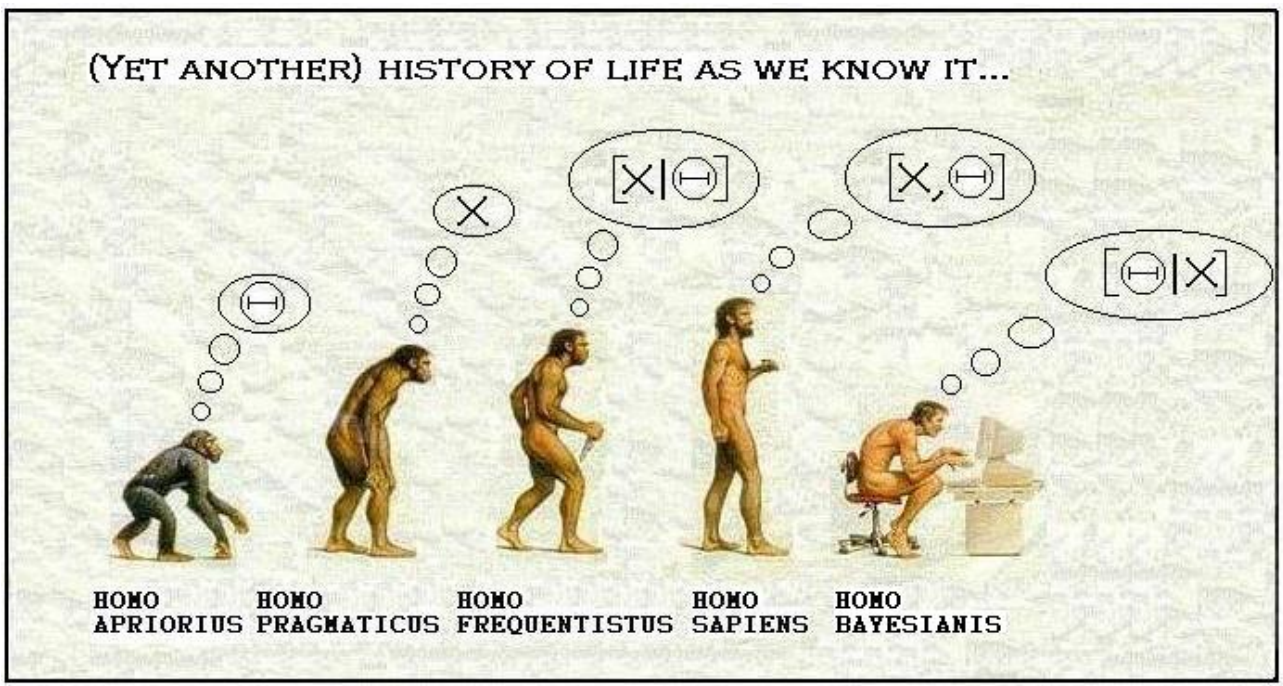
Clearly, the only way to infer the nature and geometry of surface properties from image properties is to build-in certain assumptions about the nature of the surfaces from other kinds of evidence. This requires us to consider the general problem of inference and integration of evidence.

(Surfaces and reflectance maps, con't)

Computing "shape-from-shading" requires the disambiguation of:

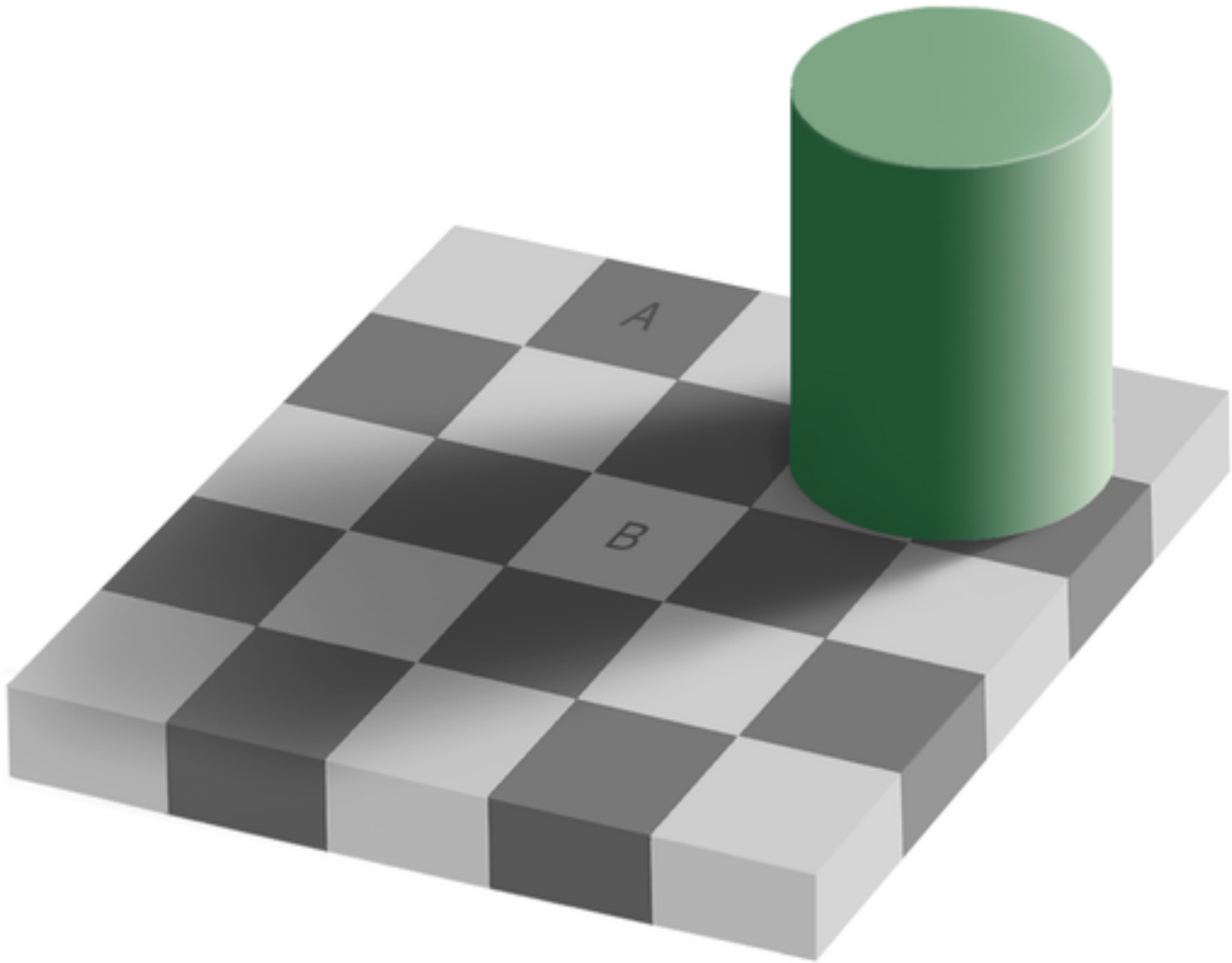
- geometry of the illuminant (e.g. is it a point source, or extended? If a point source, where is it?) Are there several light sources?
- reflectance properties of the surface. What is its reflectance map?
- geometry of the surface (its underlying shape). Are shadows cast?
- rotations of the surface relative to perspective angle and illuminant
- variations in material and surface reflectance properties across space
- variations in surface albedo ("greyness")

We must reason about hypotheses using data and assumptions:



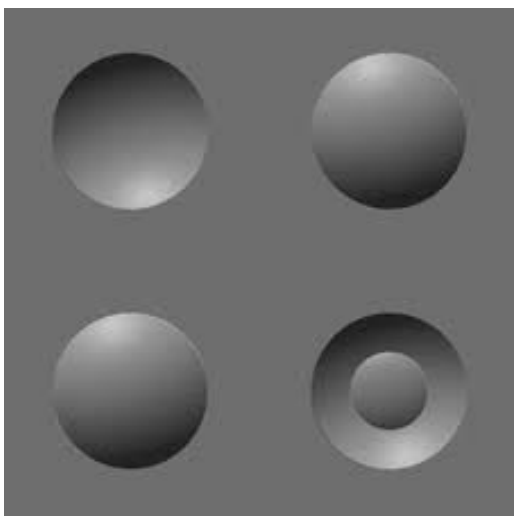
(Surfaces and reflectance maps, con't)

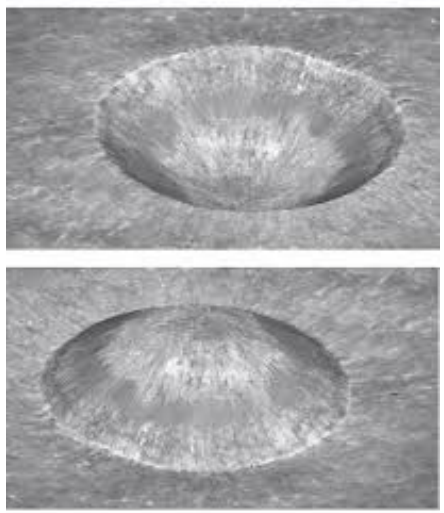
Sometimes the only consistent solution is to assume simply that the surface albedo really is different. In this image, tile A is emitting the same light as tile B. But the requirements of illumination context and shading make it impossible to see them as having the same albedo:

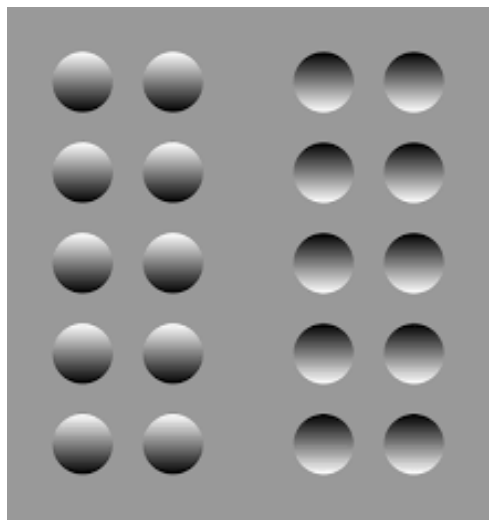


(Surfaces and reflectance maps, con't)

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 not known. One has to introduce ancillary assumptions about the surface material composition, its albedo and reflectance map, the illumination of the scene and its geometry, before such inferences become possible.







It is almost as though the assumptions (like angle of illumination) are more important than the available image data. The computational nature of the inference task then becomes one of *constraint satisfaction*. Often there are rivalrous (e.g. is it a dome or a crater?) alternative solutions:

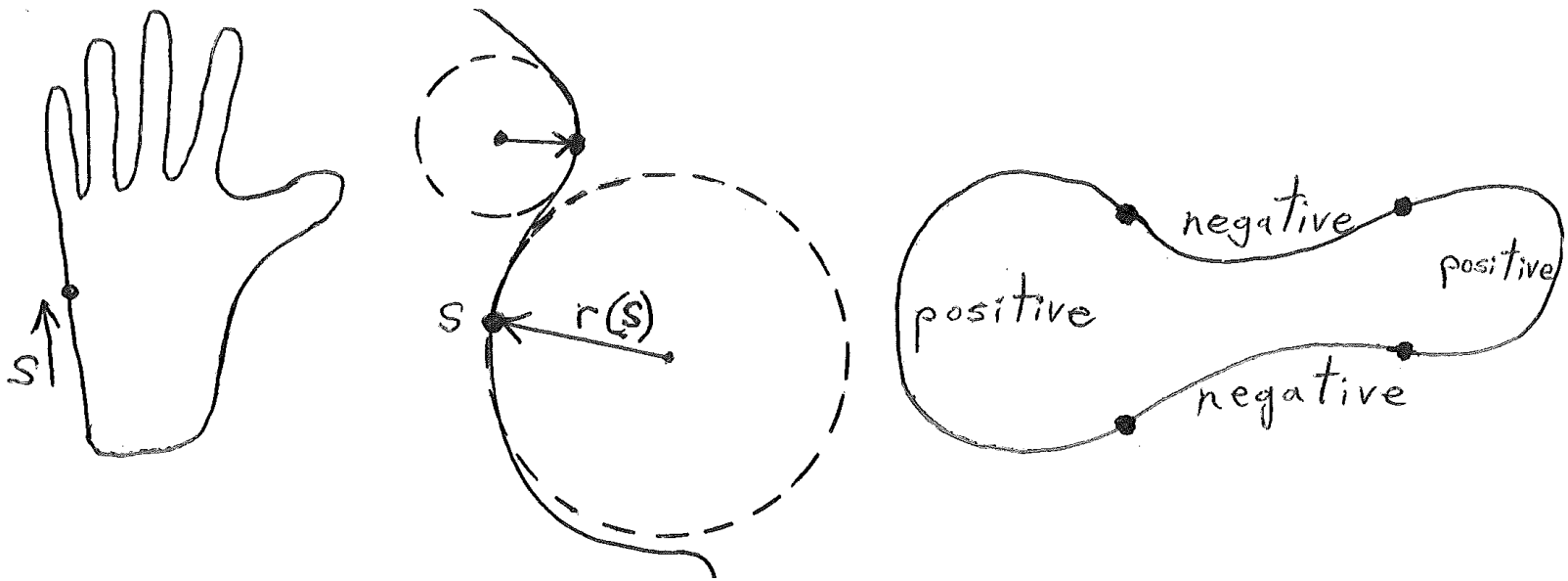
http://www.michaelbach.de/ot/fcs_hollow-face/index.html

10. Shape representation and codon shape grammars

Closed boundary contours can be represented completely by their curvature map $\theta(s)$ as a function of position s along the perimeter:

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

where the local radius of curvature $r(s)$ is defined as the limiting radius of the circle that best "fits" the contour at position s, as arc $\Delta s \to 0$. Curvature sign, $+/-$, depends on whether the circle is inside, or outside, the figure. For open contours, other conventions determine the sign. The figure's concavities are linked with minima; its convexities with maxima.



(Shape representation and codon shape grammars, con't)

The purpose of computing shape descriptions like curvature maps $\theta(s)$ (which might result from fitting active contours, for example), is to build a compact classification grammar for recognising common shapes.

By the Fundamental Theorem of Curves, a 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-reversed shapes just by changing the sign of s, so the perimeter is traversed in the opposite direction:

$$\theta(s) \rightarrow \theta(-s)$$

4. Scaling property: Changing the size of a shape just scales $\theta(s)$ by a constant (K is reciprocal to the size change factor):

$$\theta(s) \rightarrow K\theta(s)$$

(Shape representation and codon shape grammars, con't)

A visual classifier needs to recognise objects by their fundamental shapes, regardless of their orientation, position, and size (hence distance).

What is the essence of (say) a cashew nut shape, and how can it be recognised with invariance to all of those irrelevant transformations?

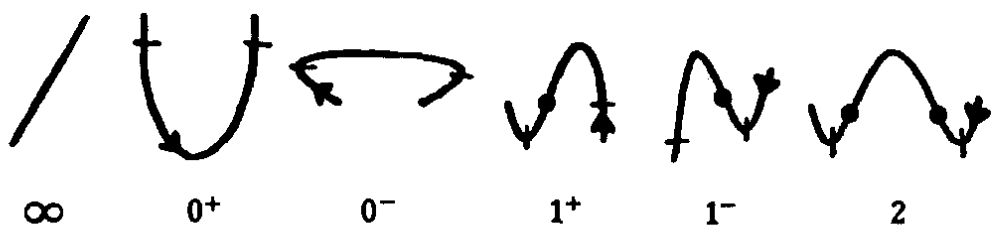
Can a compact shape description language be created that can be used for indexing into a classifier's codebook of canonical shapes?

(Shape representation and codon shape grammars, con't)

The goal is to construct an hierarchical taxonomy of closed 2D shapes, based on the extrema of curvature. Their possible combinations are very restricted by the requirement of closure, leading to a codon grammar of shapes (analogous to the ordered triples of the nucleotide bases A,G,C,T which specify the 20 amino acids).

Note that since curvature is a signed quantity (depending on whether the fitting circle is inside or outside the shape), the minimum and maximum of curvature may mean the same radius. For open contours, they depend on sign conventions and the direction of travel. We are interested in the extrema of curvature: minima, maxima, and zeroes (the inflexion points).

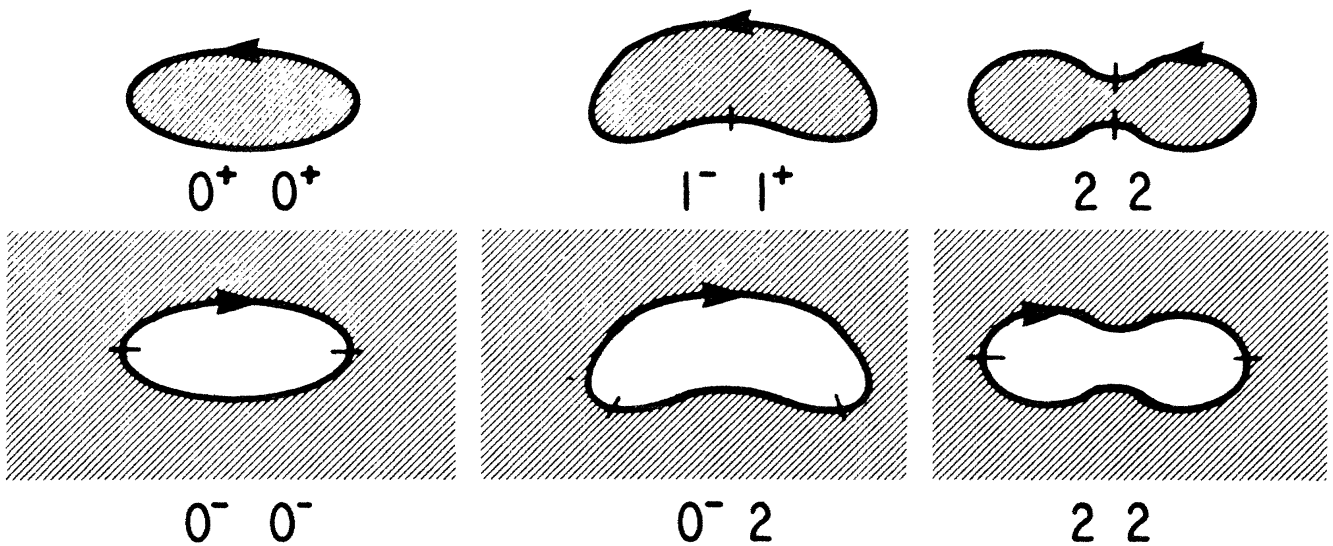
There are just six primitive codon types: all curve segments lying between minima of curvature must have 0, 1 or 2 points of zero curvature, further classified by whether a zero is encountered before ("$-$") or after ("$+$") reaching the maximum curvature in the chosen direction of traversal. Dots show zeroes of curvature (inflexions); slashes indicate the minima:



(Shape representation and codon shape grammars, con't)

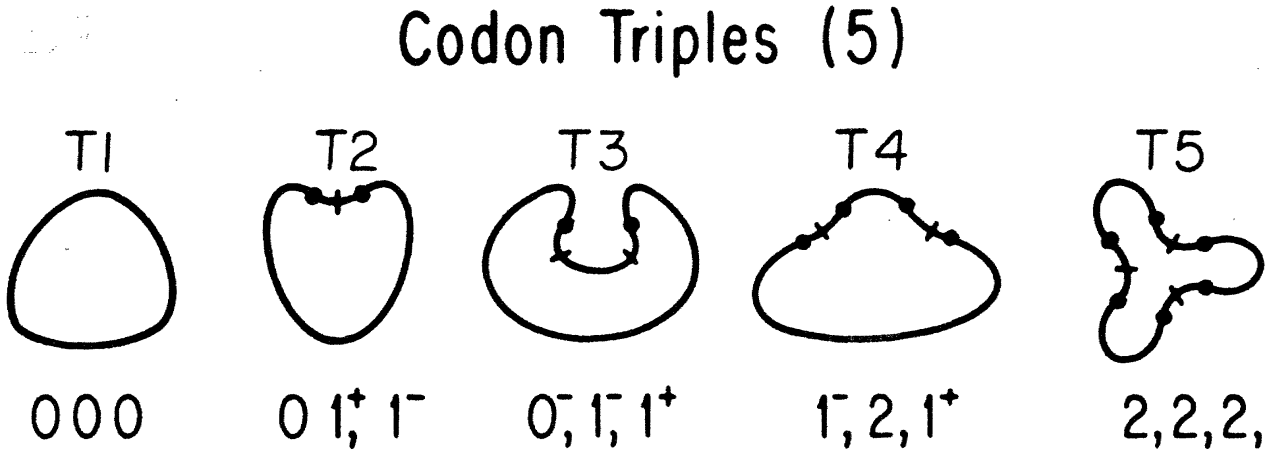
Note that because curvature is a signed quantity, the loci of its minima depend on what we take to be "figure" vs "ground". For open contours like these face profiles (alternatively Rubin's Vase profiles), if we regard "figure" as "to left", then loci of minima depend on direction of traversal:

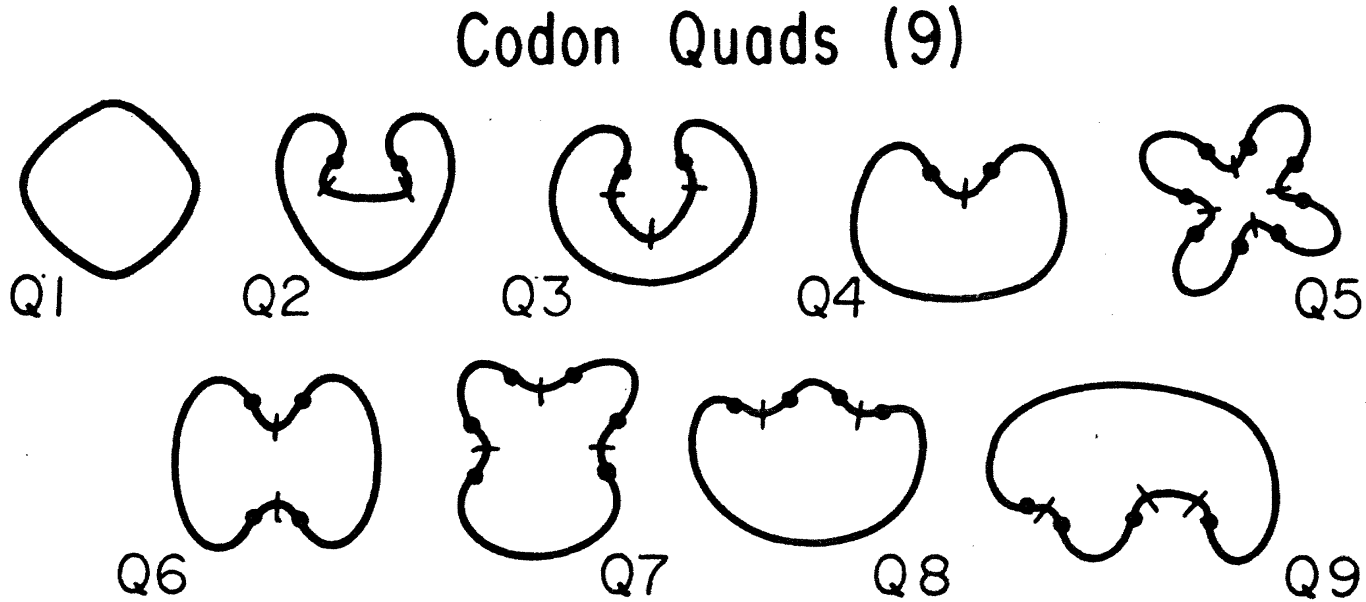
There are 3 possible Codon Pairs (string type depending on direction):



(Shape representation and codon shape grammars, con't)

There are 5 possible Codon Triples, and 9 possible Codon Quads:





(Shape representation and codon shape grammars, con't)

Constraints on codon strings for closed curves are very strong. While sequences of (say) 6 codons have $5^6 = 15,625$ possible combinations, these make only 33 generic shapes.

Ordinal relations among singular points of curvature (maxima, minima, and zeroes) remain invariant under translations, rotations, and dilations.

The inflexion (a zero of curvature) of a 3D curve is preserved under 2D projection, thereby guaranteeing that the ordinal relations among the extrema of curvature will also be preserved when projected to an image.

Thus we can acquire a very compact lexicon of elementary shapes, and we can construct an object classification algorithm as follows:

1. use active contours to fit a deformable snake to an object's outline
2. extract its codon string from its curvature map $\theta(s)$ by traversing the outline given after convergence of the active contours algorithm
3. use this codon string as an index to a labelled lexicon of shapes
4. object is then classified by shape, with invariance to many factors.

Volumetric descriptions of 3D shape

Superquadrics represent objects as the unions and/or intersections of generalized superquadric closed surfaces, which are the loci of points in (x, y, z)-space that satisfy parametric equations of this form:

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

Spheres have $(\alpha, \beta, \gamma) = (2, 2, 2)$ and $A = B = C$. Other examples:

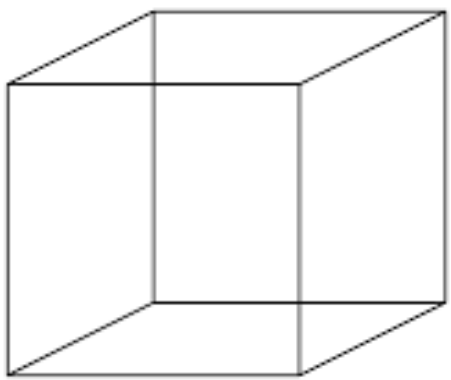
- cylinders: $(\alpha, \beta, \gamma) = (2, 2, 100)$ and $A = B$
- rectangular solids: $(\alpha, \beta, \gamma) = (100, 100, 100)$
- prolate spheroids (shaped like zeppelins): $(\alpha, \beta, \gamma) = (2, 2, 2)$ and (say) $A = B$ but $C < (A, B)$
- oblate spheroids (shaped like tomatoes): $(\alpha, \beta, \gamma) = (2, 2, 2)$ and (say) $A = B$ but $C > (A, B)$

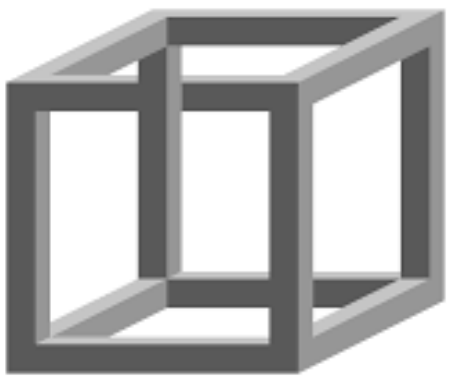
Rotations of such objects in 3D produce cross-terms in (xy, xz, yz). Parameters (A, B, C) determine object dimensions. Origin-centred.

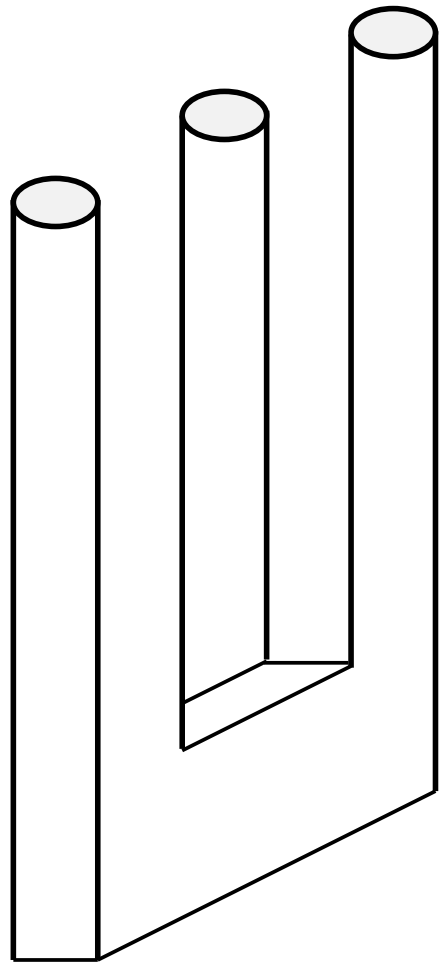
These simple, parametric models for solids, augmented by Boolean relations for conjoining them, allow the generation of object-centered, "volumetric" descriptions of many objects (instead of an image-based description) by just listing parameters $(\alpha, \beta, \gamma, A, B, C)$ and relations, rather like the codon descriptors for closed 2D shapes.

11. Vision as model building

- role of context in determining a model
- percepts as hypotheses generated for testing
- rivalrous and paradoxical percepts, and visual illusions: "bugs" or "features" of a system?

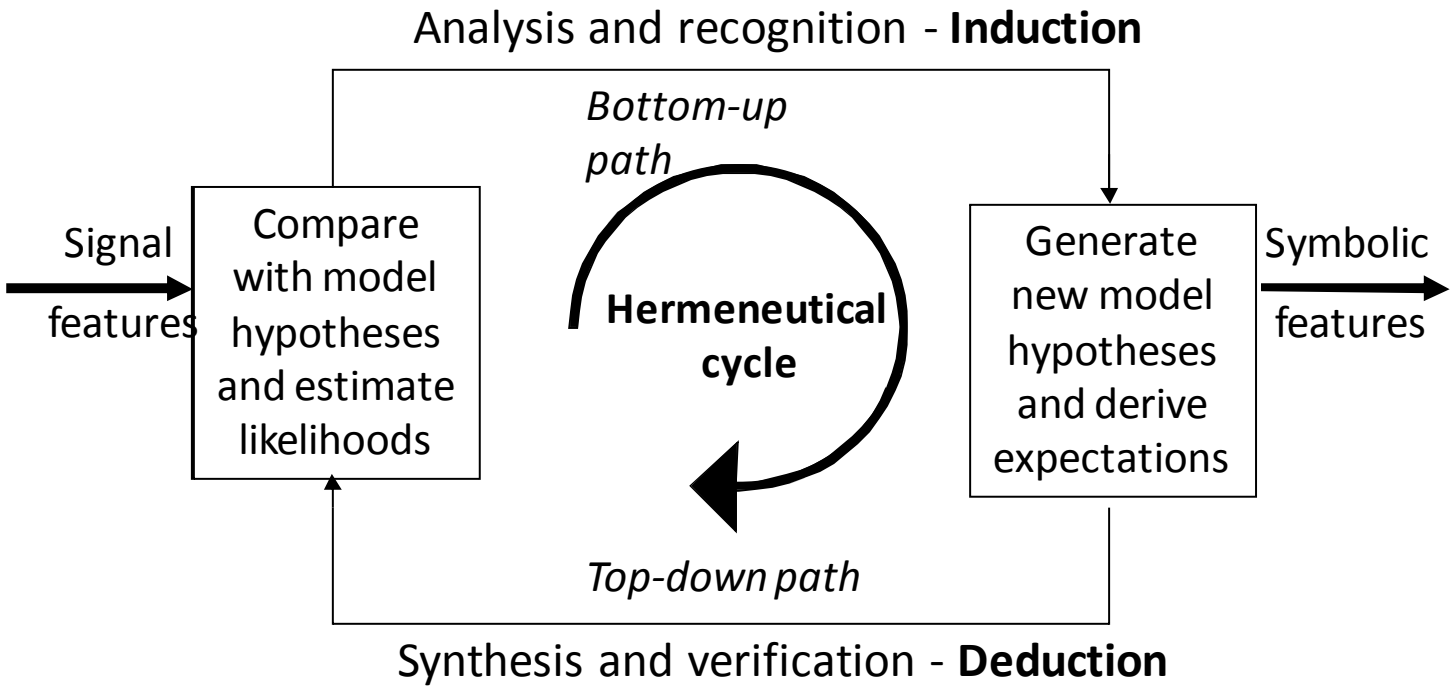






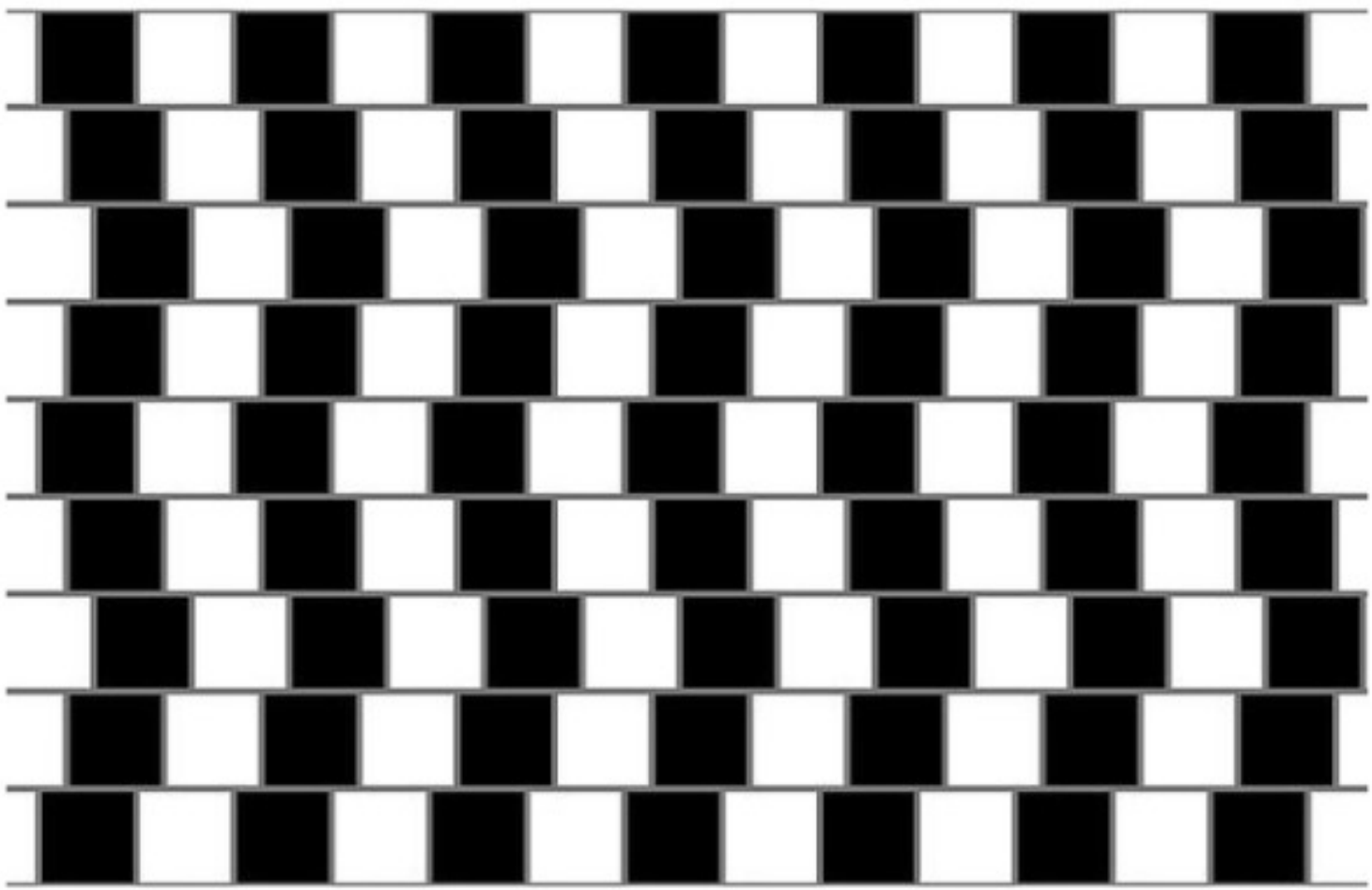
Vision as perceptual inference and hypothesis testing

- Low-level visual percepts, built from extracted features, must be iteratively compared with high-level models to derive hypotheses about the visual world
- This iterative cycle of model-building for hypothesis generation and testing is sometimes called the hermeneutical cycle
- It fits the key anatomical observation that mammalian brains have massive feedback projections from the visual cortex back down to the thalamus, meeting the upcoming data stream from the eyes

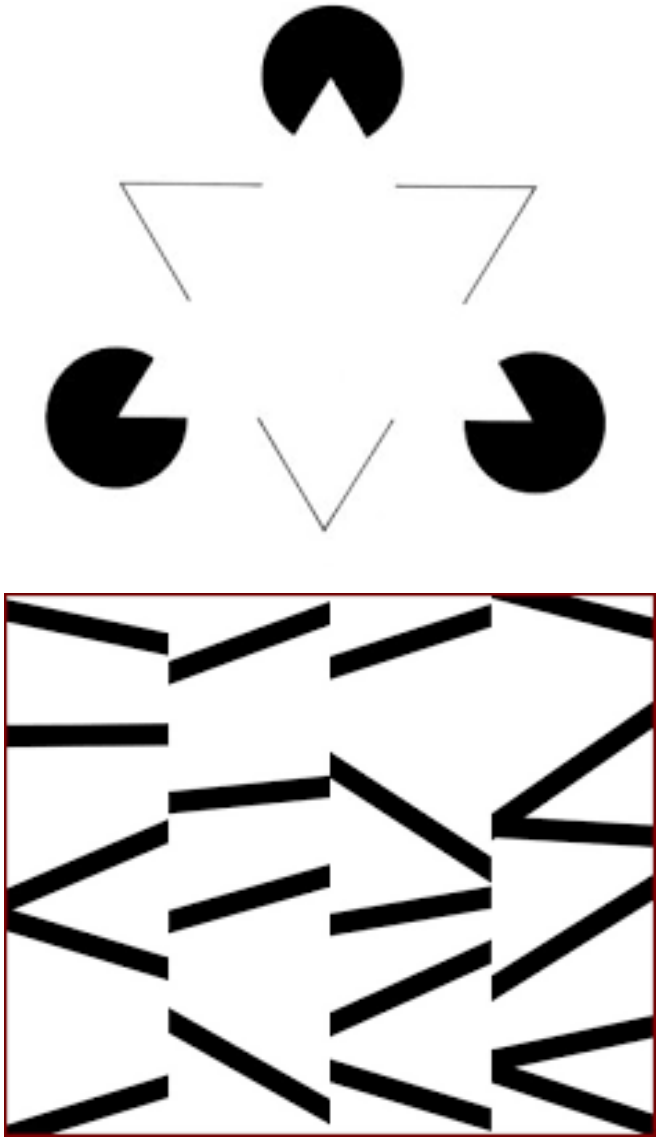


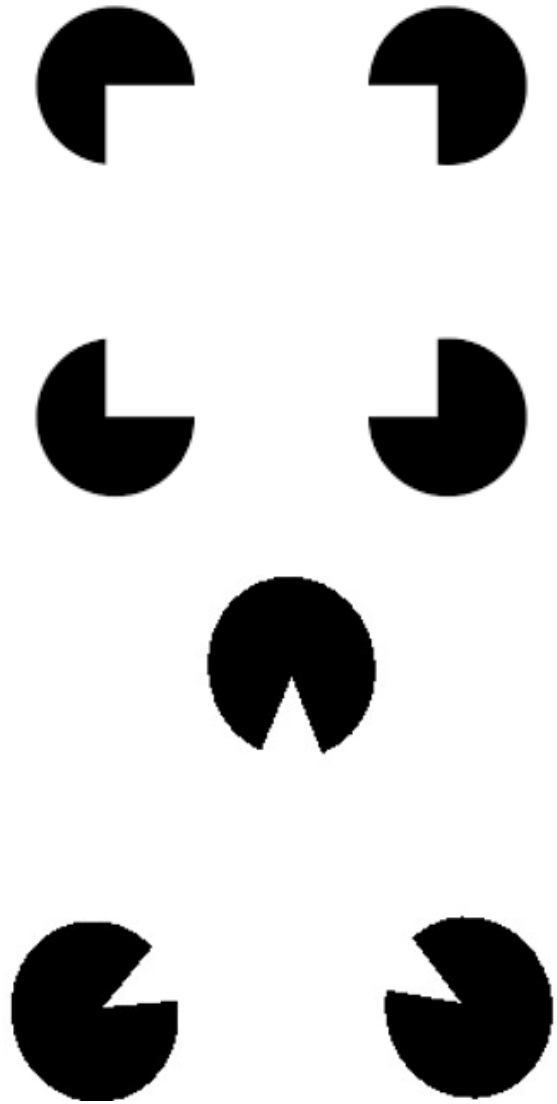
12. Lessons from visual illusions, neural trauma, & deficits

- Normal human vision is often not veridical. Illusions are standard.
- Illusions can reveal top-down processing; the role of expectation; and interactions between cooperative and competitive neural processes.
- In the "cafe wall illusion" below, all long lines are actually parallel.
- Are illusions features or bugs? Should we design them into systems?



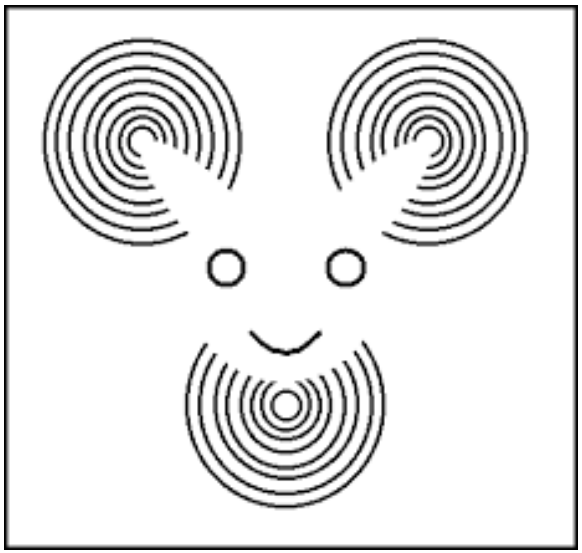
Neurones (in cats) actually respond to illusory contours

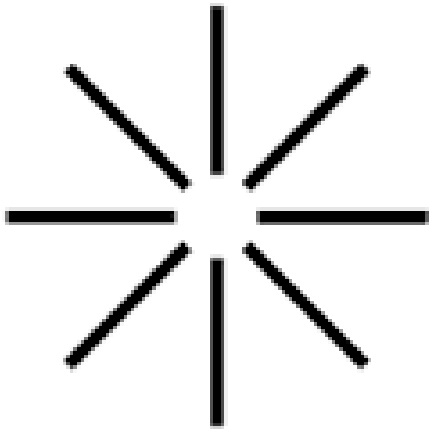


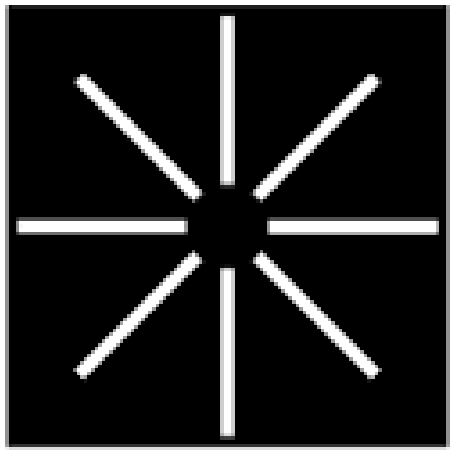


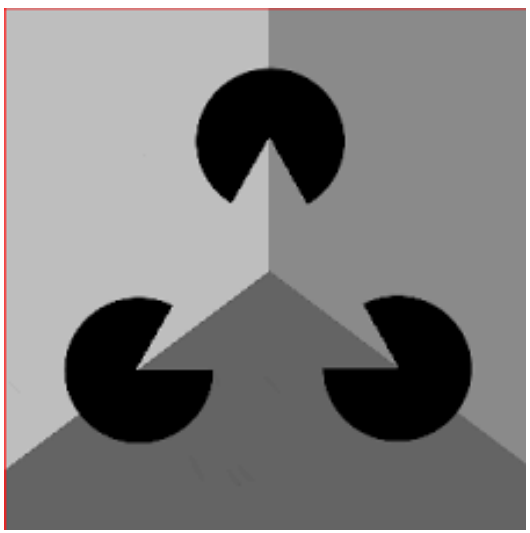
"Illusory contours and cortical neuron responses", *Science* **224** (1984), pp. 1260-1262.

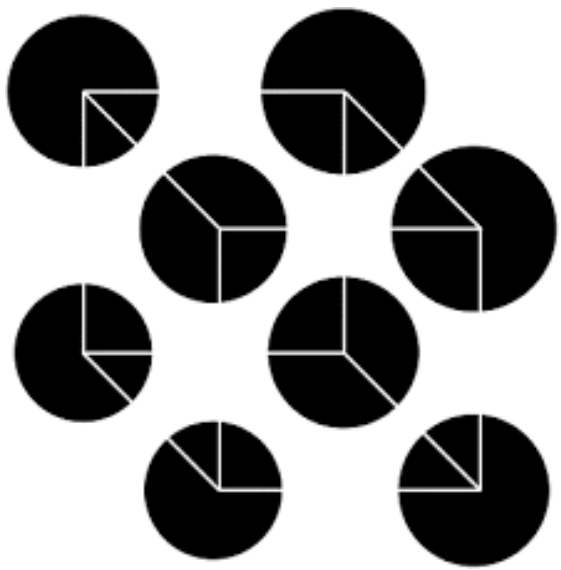
Illusory contours can even drive some high-level inferences











Lessons from neurological trauma and deficits

Strokes and battlefield injuries sometimes have astonishing consequences, with aphasias and agnosias indicating highly specialised brain areas.

- *Facial prosopagnosia:* lost ability to recognise faces. Vision appears normal otherwise, but faces cease to be represented or processed.
- *Achromatopsia:* cortical loss of colour vision, but "black-and-white" (achromatic) vision is apparently normal.
- *Astereognosia:* loss of ability to perceive three-dimensionality.
- *Simultanagnosia:* inability to perceive simultaneously more than one thing at a time (e.g. multiple elements in a display).
- *Neglect* and *hemi-inattention syndromes*: one side of any object is always neglected. Such patients dress themselves only on (say) their right side, and always bump into things with their left side; and will draw a clock face with all the numbers 1 - 12 in the right half only.
- *Xanthopsia:* perception that all objects are covered with gold paint.

What sort of "computer" is the brain, that it can display *these* types of faults when traumatised? What do these phenomena reveal about the nature of the brain's architecture, data structures, and algorithms?

13. Bayesian inference in vision

It is almost impossible to perform most computer vision tasks in a purely "bottom-up" fashion. The data are just too impoverished by themselves to support the task of object recognition

(Bayesian inference in vision, con't)

The Bayesian view focuses on the use of priors, which allow vision to be steered heavily by *a priori* knowledge about the world and the things which populate it.

For example, probabilistic priors can express the notions that:

- some events, objects, or interpretations are much more probable than others
- matter cannot just disappear, but it does routinely become occluded
- objects rarely change their actual surface colour
- uniform texturing on a complex surface shape is a more likely interpretation than highly non-uniform texturing on a simple or planar surface
- a rigid rotation in three dimensions is a "better explanation" for deforming boundaries (if consistent with them) than wild actual boundary deformations in the object itself

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.

(Bayesian inference in vision, con't)

Bayes' rule is a formalism for combining prior knowledge or beliefs with empirical observations. It is at once a theory of explanation, a method for drawing inferences from data, a procedure for the integration of evidence, and a protocol for decision-making.

If H represents an 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:

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

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

(Bayesian inference in vision, con't)

- Bayes' rule specifies the formal procedure for calculating inferences $p(H|D)$, given the observations, the unconditional probabilities, and the prior expert knowledge $p(D|H)$
- It thereby offers a clean and simple interface between a knowledge base and incoming visual data
- A key feature is that it provides a formal 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* estimate $p(H|D)$ as the new *prior* $p(H)$ for interpreting the next set of data
- Thus we learn from visual data and experience, and we can build up visual knowledge about a domain of the world: we learn to see
- In AI, this aspect is important because it allows the systematic and real-time construction of interpretations that can be continuously updated as more data arrive in a time series, such as in a sequence of video or of spoken sounds that we wish to understand

Statistical decision theory

In many applications, we need to perform pattern classification on the basis of some vector of acquired features from a given object or image.

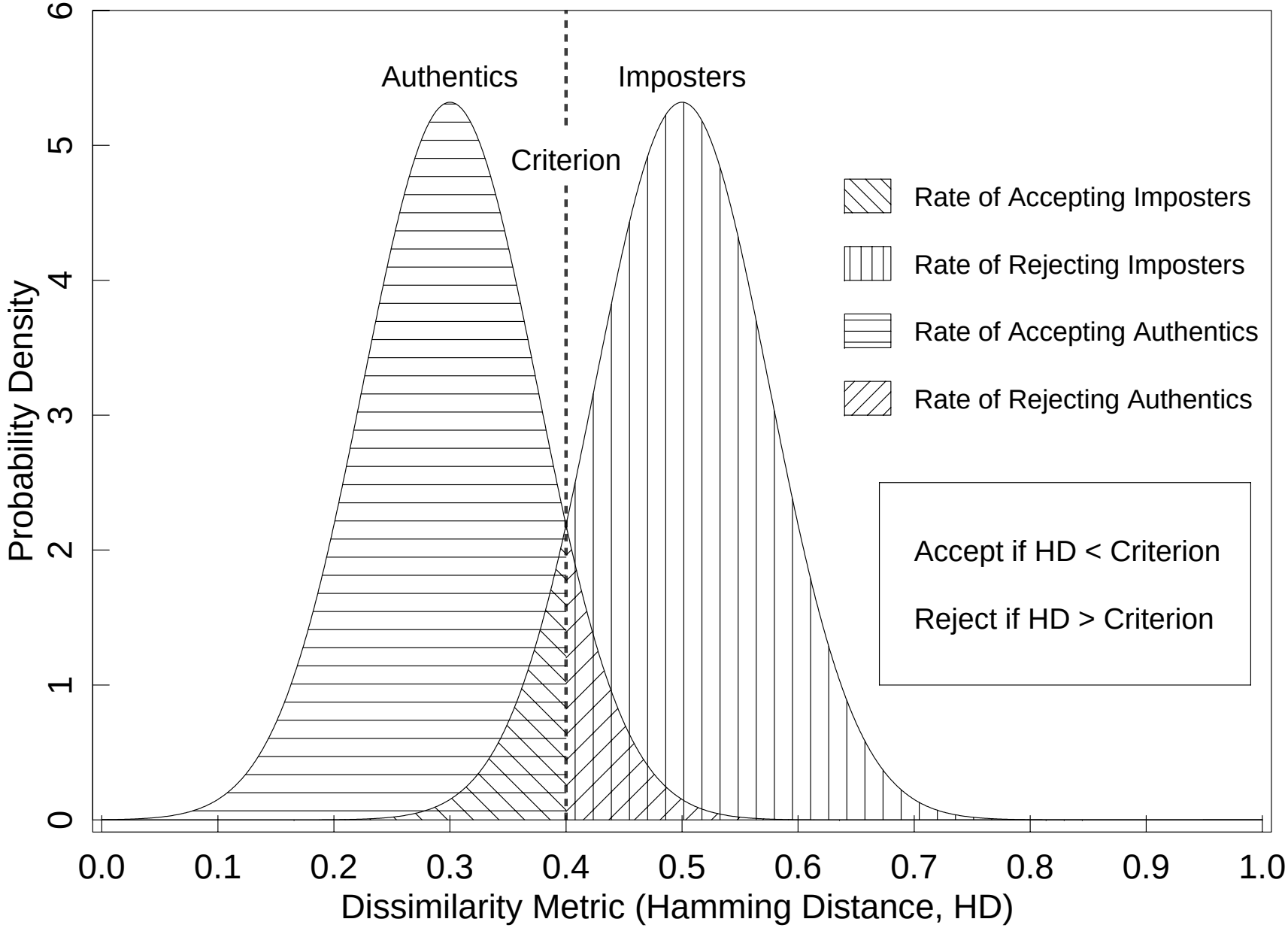
The task is to decide whether or not this feature vector is consistent with a particular class or object category. Thus the problem of classification amounts to a "same / different" decision about the presenting feature vector, compared with vectors characteristic of certain object classes.

Usually there is *some* similarity between "different" patterns, and *some* dissimilarity between "same" patterns. The four possible combinations of "ground truths" and decisions creates a decision environment:

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

We would like to maximize the probability of outcomes 1 and 4, because these are correct decisions. We would like to minimize the probability of outcomes 2 and 3, because these are incorrect decisions

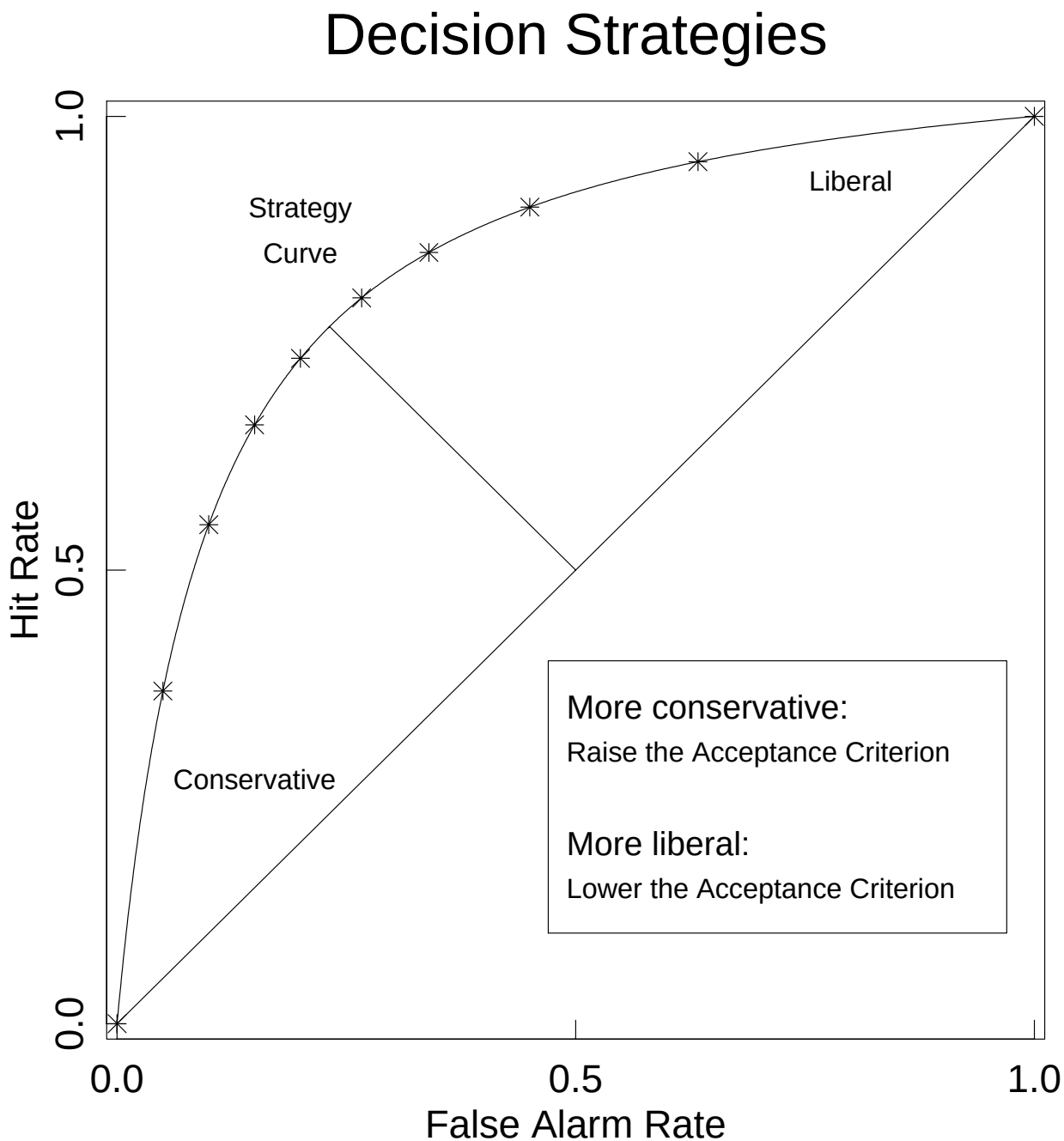
Statistical Decision Theory



(Statistical decision theory, con't)

- In the two-state decision problem, the feature vectors or data are regarded as arising from two overlapping probability distributions
- They might represent the features of two object classes, or they might represent the similarity scores for "same" vs "different"
- When a decision is made, based upon the observed similarity and some acceptability threshold, the probabilities of the four possible outcomes can be computed as the four cumulatives under these two probability distributions, to either side of the decision criterion
- These four probabilities correspond to the shaded areas in last figure
- The computed error probabilities can be translated directly into a confidence level which can be assigned to any decision that is made
- Moving the decision criterion (dashed line) has coupled effects:
  - Increasing the "Hit" rate also increases the "False Alarm" rate
  - Decreasing the "Miss" rate also decreases the "Correct Reject" rate
- These dependencies map out the Receiver Operating Characteristic
- Each point ($*$) on the ROC curve (next fig.) represents a particular choice for the decision criterion, or threshold of acceptance

Receiver Operator Characteristic ("ROC curve")



(Statistical decision theory, con't)

Obviously we would like the ROC curve to be as "bowed" as possible, approaching into the upper left corner, as that maximises the Hit Rate and minimises the False Alarm Rate.

Regardless of where our decision criterion is placed, the fundamental decidability of the decision task (or the detectability in a detection task) is measured by the quantity d', which is monotonically related to the length of the "arrow" in the "bow" (how bowed the ROC curve is):

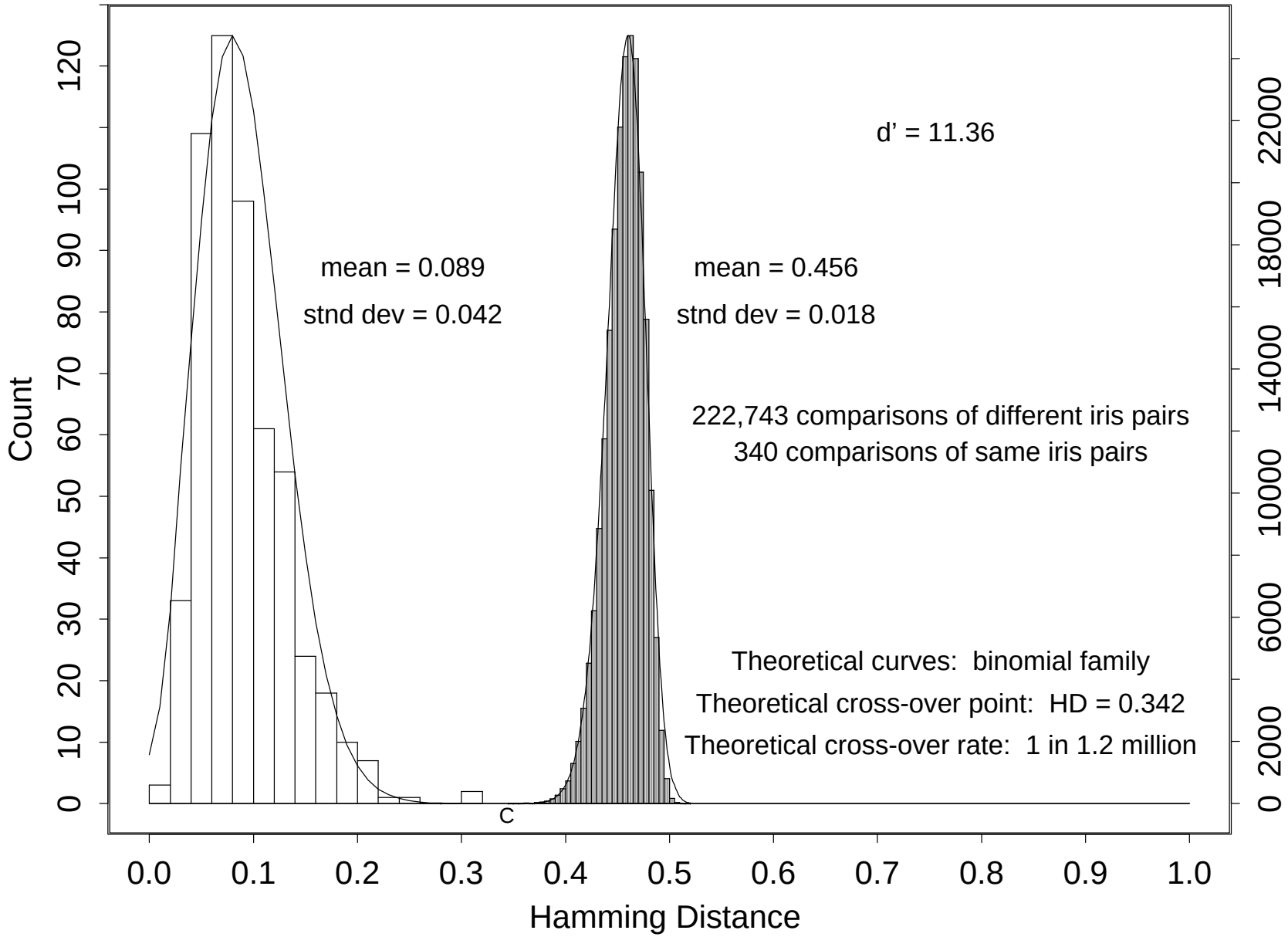
$$d' = \frac{|\mu_2 - \mu_1|}{\sqrt{\frac{1}{2}(\sigma_2^2 + \sigma_1^2)}}$$

where the two distributions are characterized by their means μ_1 and μ_2 and their standard deviations σ_1 and σ_2. The metric d' is also called discriminability. It is related to other σ-normalisations, such as Z-scores.

An improvement in d' can result either from pushing the distributions further apart, or from making one or both of them narrower. The bigger d' is, the better; a pattern recognition problem with high decidability will have a large d', so the ROC curve approaches the upper-left corner.

(Statistical decision theory, con't)

Decision Environment for Iris Recognition: same vs different eyes

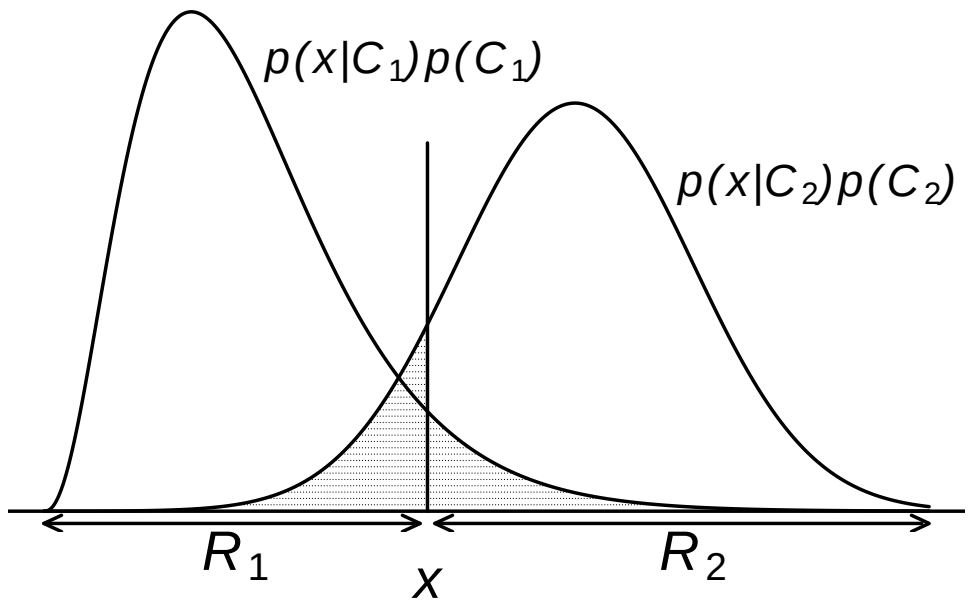


(Statistical decision theory, con't)

Decidability $d' \geq 3$ is normally considered good. The distributions shown originally to illustrate had $d' = 2$. The empirical ones for iris recognition (previous figure) had $d' \approx 11$.

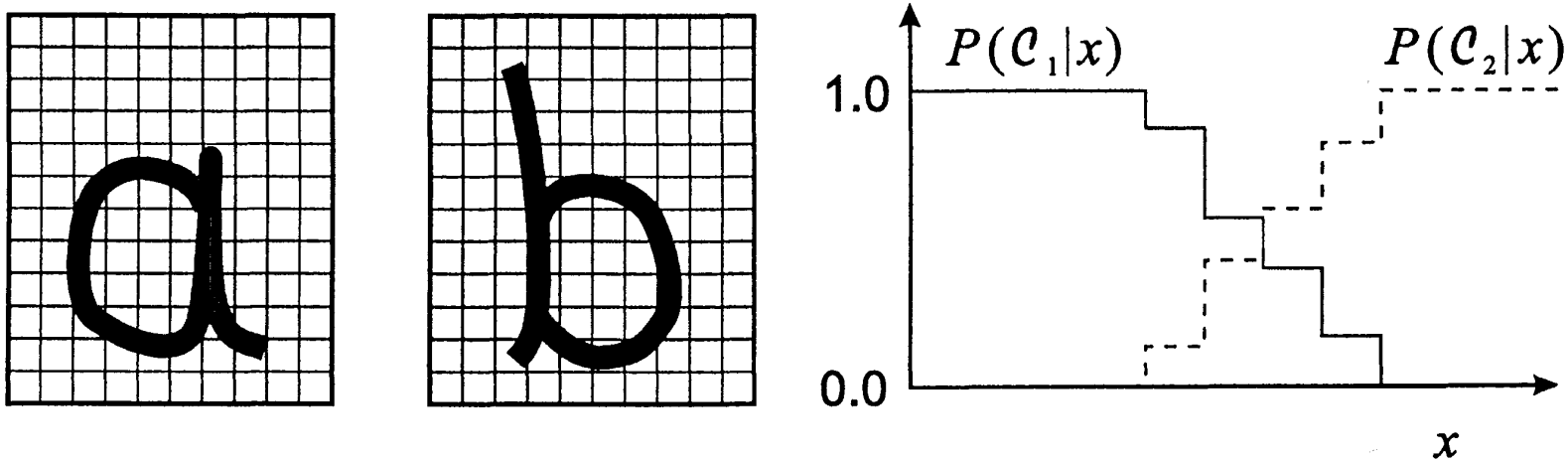
Because reliability of pattern recognition depends on the between-class variance being larger than the within-class variance, R. Fisher defined the "separation between two distributions" as the ratio of their between-class variance to their within-class variance. This definition is related to d'.

Another metric is the total area under the ROC curve, which ideally $\rightarrow 1$. Other relevant metrics include the total probability of error for a chosen decision criterion, as illustrated by the combined shaded areas below:



Bayesian pattern classifiers

Consider a two-class pattern classification problem, such as OCR (optical character recognition) involving only two letters, a and b. We compute some set of features x from the image data, and we wish to build a Bayesian classifier that will assign a given pattern to one of two classes, $C_1 \equiv a$ or $C_2 \equiv b$, corresponding to the two letter instances.



Whatever are the extracted features x (perhaps they are as simple as 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 classes. In general, these histograms will overlap, but clearly the smaller x is, the more likely it is that this sample came from class C_1, other things being equal.

(Bayesian pattern classifiers, con't)

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, the letter class is more likely to have been C_1 than C_2?

No. As Bayesians we must take into account the baseline rates. Define the prior probabilities $P(C_1)$ and $P(C_2)$ as their two relative frequencies (summing to 1).

If we had to guess which character had appeared without even seeing it, we would always just guess the one with the higher prior probability.

For example, since in fact an 'a' is about 4 times more frequent than a 'b' in English, and these are the only two options in this two-class inference problem, we would set the priors $P(a) = 0.8$ and $P(b) = 0.2$ then.

(Bayesian pattern classifiers, con't)

- For each class separately, we can measure how likely any particular feature sample value x will be, by empirical observation of examples
- (Note that this requires knowing the "ground truth" of examples)
- This gives us $P(x|C_k)$ for all the classes C_k
- We get the unconditional probability $P(x)$ of any measurement x by summing $P(x|C_k)$ over all the classes, weighted by their frequencies:

$$P(x) = \sum_k P(x|C_k)P(C_k)$$

- Now we have all the terms needed to compute posterior probabilities $P(C_k|x)$ of class membership, given some data observation x, taking into account the priors $P(C_k)$ and the "class conditional likelihoods" $P(x|C_k)$ of the observations x:

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

(Bayesian pattern classifiers, con't)

Thus we have a principled, formal way to perform pattern classifications on the basis of available data and our knowledge of class baseline rates, and how likely the data would be for each of the classes.

We can minimise the total probability of misclassification if we assign each observation x to the class with the highest posterior probability.

Assign x to class C_k if:

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

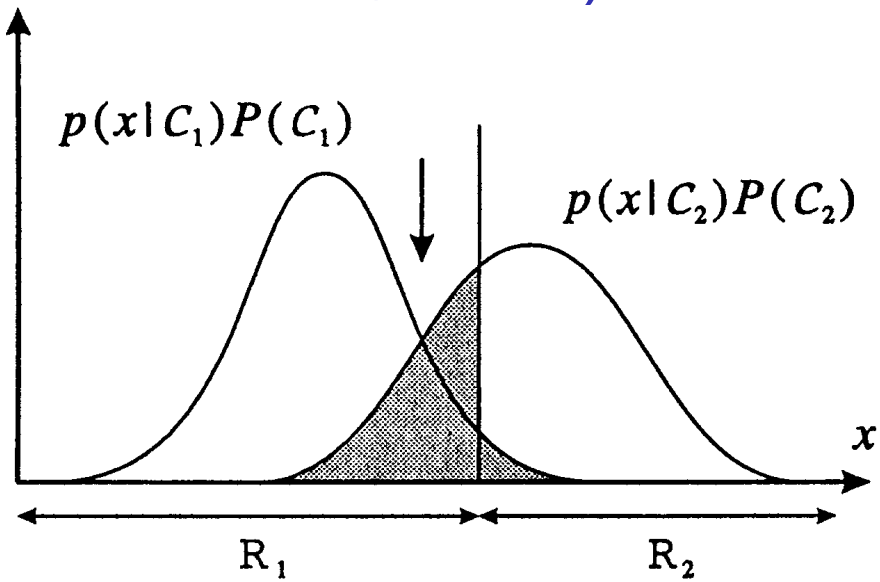
Since the denominator $P(x)$ in Bayes' Rule is independent of C_k, we can rewrite this minimum misclassification criterion simply as:

Assign x to class C_k if:

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

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

(Bayesian pattern classifiers, con't)



Because the costs of the two different types of errors are not always equal, 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 we choose is instead as indicated by the vertical line, so R_1 and R_2 are the regions of x on either side of it, then the total probability of error (which is the total shaded area) is:

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

14. Discriminant functions and decision boundaries

If we construct some set of functions $y_k(x)$ of the data x, 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 loci in the (normally multi-dimensional) data $\mathbf{x}$ at which $y_k(\mathbf{x}) = y_j(\mathbf{x})$.

Natural discriminant functions to choose are the posterior probabilities:

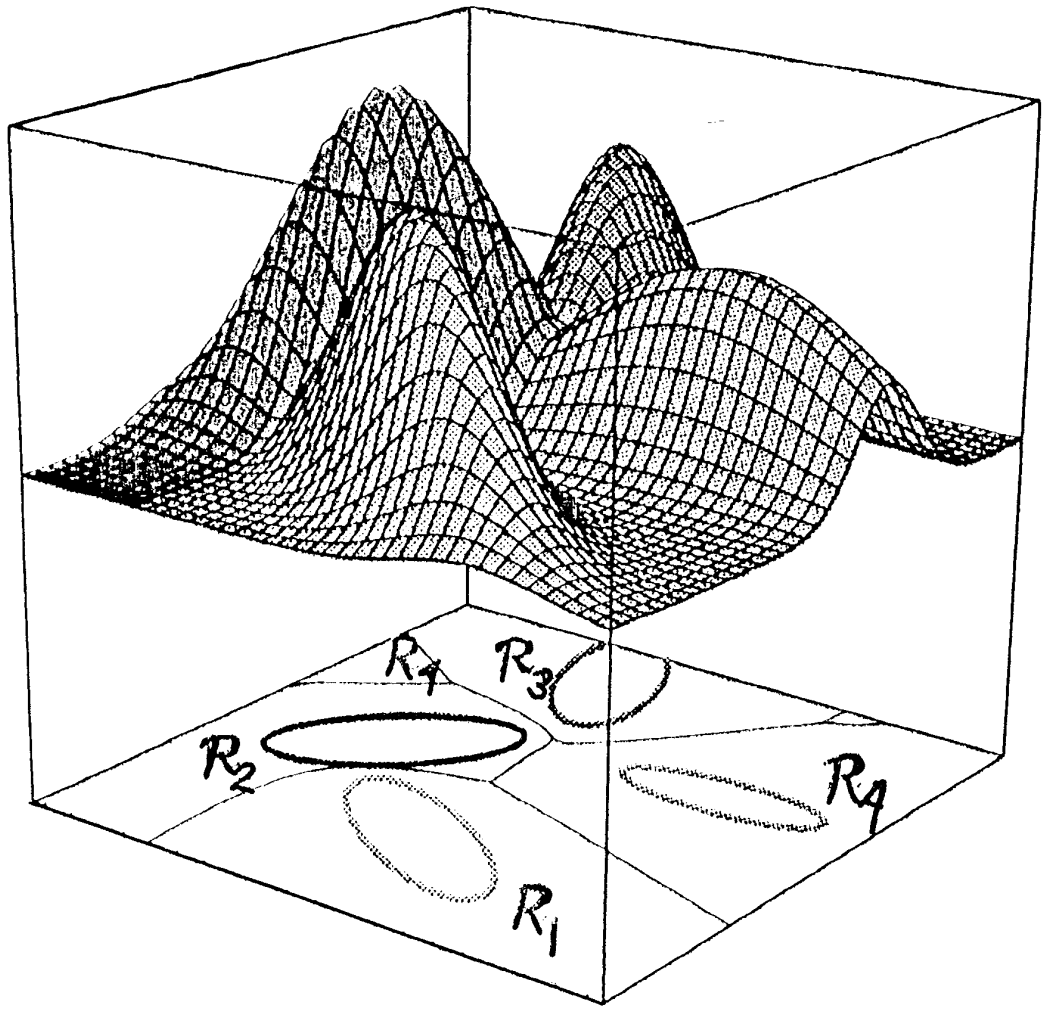
$$y_k(x) = P(C_k|x)$$

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

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

(Discriminant functions and decision boundaries, con't)

This figure shows 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. (Note how the areas corresponding to decision region R_4 are not simply connected.)



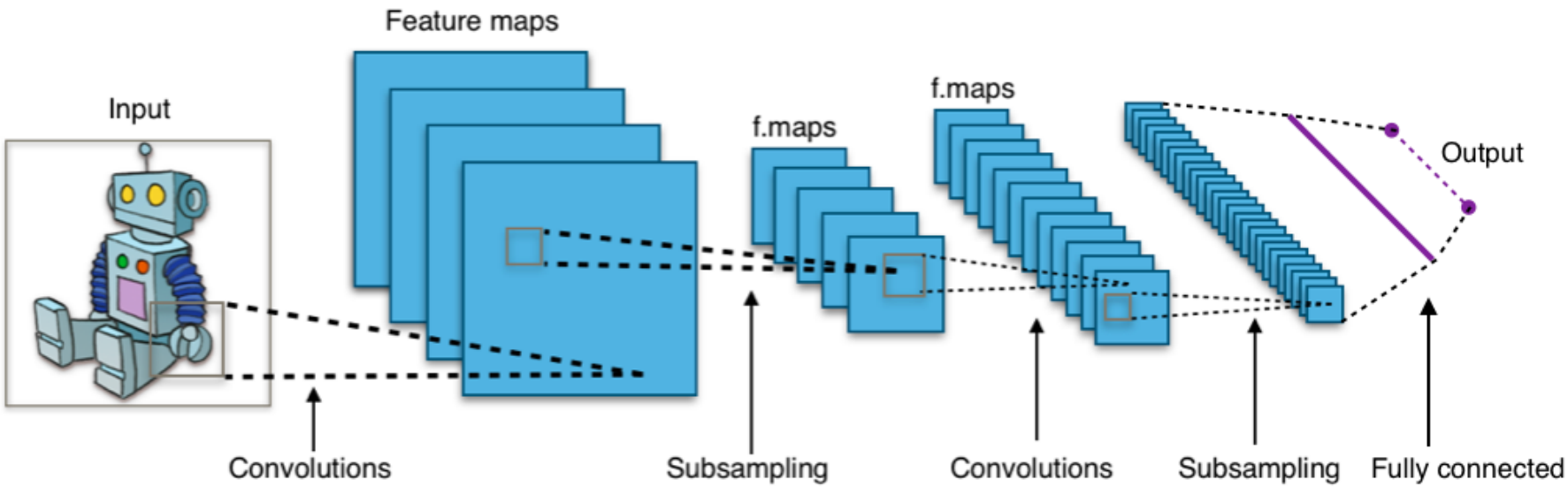
15. Discriminative versus generative methods in vision

- Discriminative methods learn a function $y_k(x) = P(C_k|x)$ that maps input features x to class labels C_k. They require large training data covering all expected kinds of variation. Examples of such methods:
  - artificial neural networks
  - support vector machines
  - boosting methods
  - linear discriminant analysis

- Generative methods learn a likelihood model $P(x|C_k)$ expressing the probability that data features x would be observed in instances of class C_k, which can then be used for classification using Bayes' Rule.
- Generalise well and need less training data, but models get complex
- Popular for tasks such as analysis and synthesis of facial expressions
- Generative models have predictive power as they allow the generation of samples from the joint distribution $P(x, C_k)$. Examples include:
  - probabilistic mixture models
  - most types of Bayesian networks
  - active appearance models
  - Hidden Markov models, Markov random fields

Convolutional neural networks

- Feedforward artificial neural networks, inspired by the visual cortex
- Perform image classification using multiple layers of small collections of neurons, having "receptive fields" in the image
- Tiling and overlapping of outputs aim to achieve shift invariance
- Often include pooling layers, convolutional layers, fully connected layers, and point non-linearities in or after each layer
- Use little pre-processing; filters learned without human intervention
- Output is a classification decision, with robust invariances over image input transformations (e.g. variations in handwritten characters)

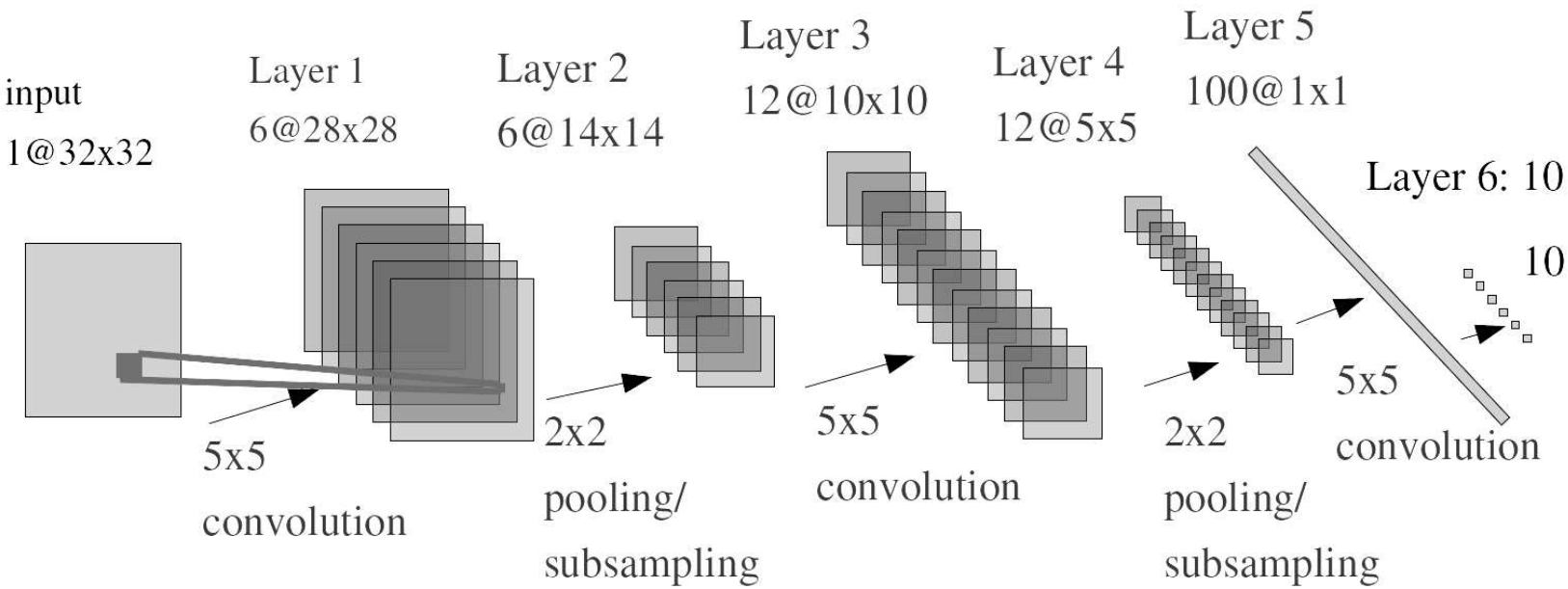


Example: convolutional neural network for OCR (LeCun)

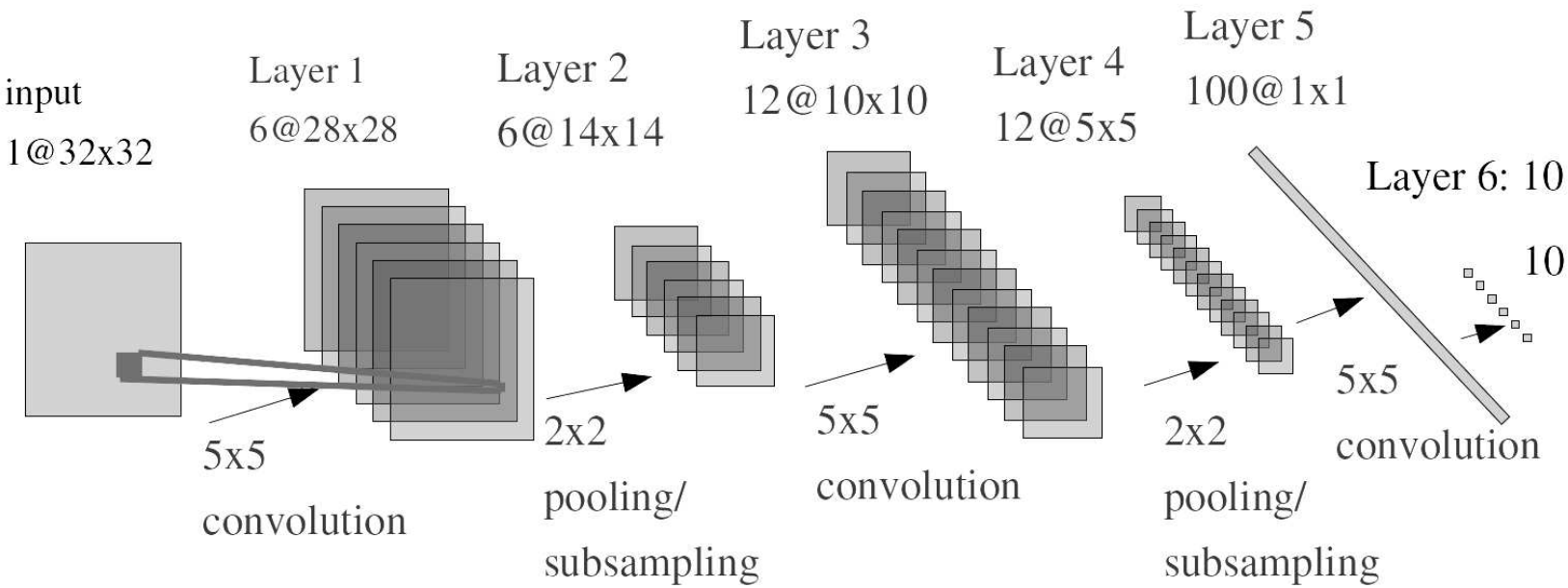
Optical Character Recognition systems have many applications:

- postal sorting, bank cheque routing
- automated number plate recognition
- book and manuscript digitisation
- text-to-speech synthesis for the blind
- handwriting recognition for portable device interfaces

Handwritten fonts require methods from Machine Learning to cope with all writing variations (size, slant, stroke thickness), distortions, and noise. A classic convolutional NN for OCR was developed by Yann LeCun:



(Example: convolutional neural network for OCR, con't)



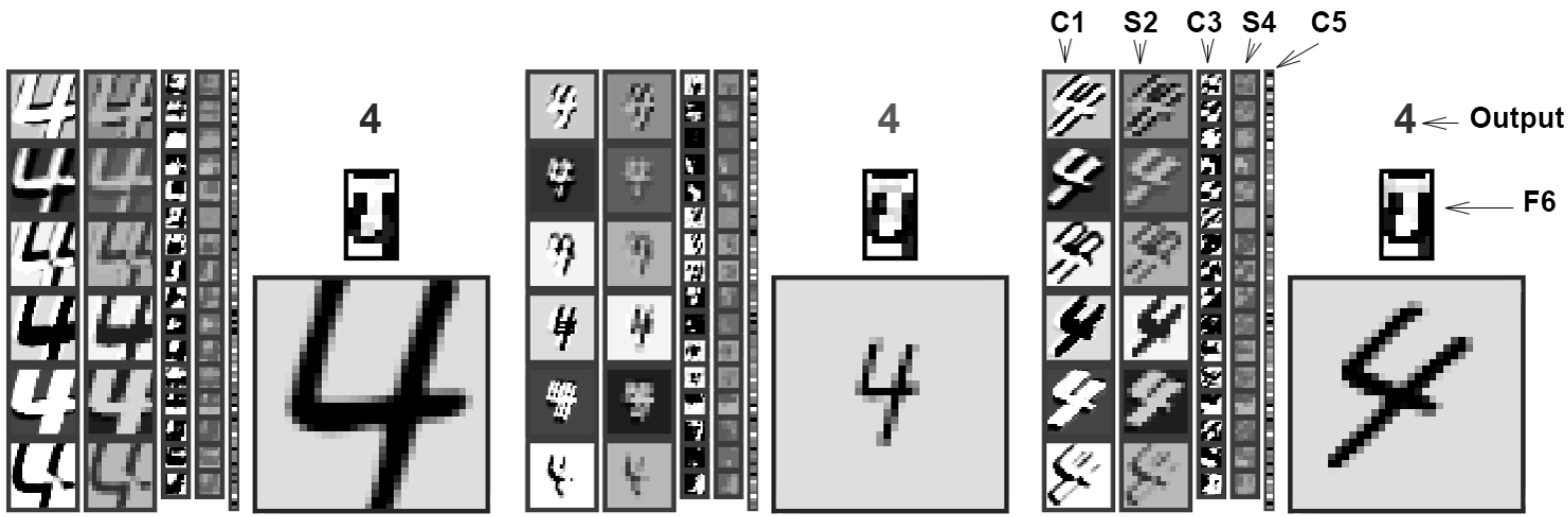
- Input is a 32×32 pixel image, containing some digit or character
- In the training phase, 100,000s of examples of each target are used
- Training is supervised back-propagation: target output is set to $+1$, all others to -1. Errors back-propagate to adaptable feature maps
- Neurons in a feature map have 5×5 kernels, convolved with input
- Trained to extract a particular visual feature, regardless of position
- Subsequent feature maps achieve size, slant, and style invariances
- Neurons in the final layer identify the input as one of the targets

(Example: convolutional neural network for OCR, con't)

The output o_{ij} of each neuron at position (i, j) applies a nonlinear (e.g., hyperbolic tangent) activation function f_{act} to the sum of its input pixels times its trained weights w_{mn}, added to another (trained) bias term w_0:

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

This figure illustrates three different handwritten instances of the digit 4 being recognised by this CNN. The smaller images show outputs of the convolutional (C) and subsampling (S) feature maps at different layers of the network.



More examples are shown at: http://yann.lecun.com/exdb/lenet/

16. Face detection, recognition, and interpretation

Some variations in facial appearance (L.L. Boilly: *Réunion de Têtes Diverses)*

(Face detection, recognition, and interpretation, con't)

Detecting faces and recognising their identity is a "Holy Grail" problem in computer vision. It is difficult for all the usual reasons:

- Faces are surfaces on 3D objects (heads), so facial images depend on pose and perspective angles, distance, and illumination
- Facial surfaces have relief, so some parts (e.g. noses) can occlude other parts. Hair can also create random occlusions and shadows
- Surface shape 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, whereas 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 or lip movements; eyebrows and winks). We have 7 pairs of facial muscles. People use their faces as communicative organs of expression
- People put things on their faces (e.g. glasses, cosmetics, cigarettes), change their facial hair (moustaches, eyebrows), and age over time

(Face detection, recognition, and interpretation, con't)

Classic problem: within-class variation (same person, different conditions) can exceed the between-class variation (different persons).

These are different persons, in genetically identical (monozygotic) pairs:

(Face detection, recognition, and interpretation, con't)

Classic problem: within-class variation (same person, different conditions) can exceed the between-class variation (different persons).

Persons who share 50% of their genes (parents and children; first-degree double cousins) sometimes look almost identical (apart from age cues):

(Face detection, recognition, and interpretation, con't)

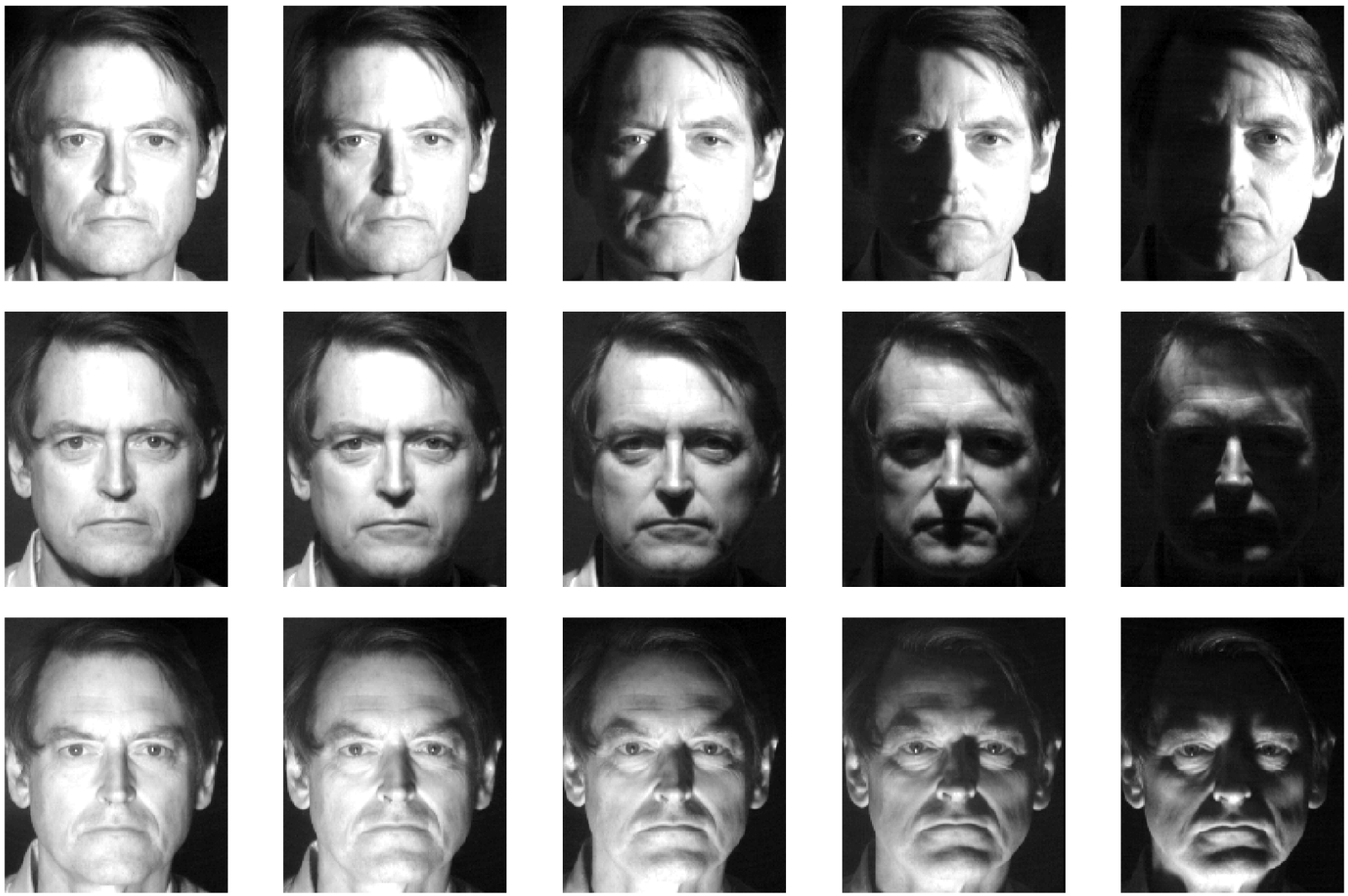
Classic problem: within-class variation (same person, different conditions) can exceed the between-class variation (different persons).

...and these are *completely unrelated people, in Doppelgänger pairs:*

(Face detection, recognition, and interpretation, con't)

Classic problem: within-class variation (same person, different conditions) can exceed the between-class variation (different persons).

Same person, fixed pose and expression; varying illumination geometry:



(Face detection, recognition, and interpretation, con't)

Classic problem: within-class variation (same person, different conditions) can exceed the between-class variation (different persons).

Effect of variations in pose angle (easy and hard), and distance:

(Face detection, recognition, and interpretation, con't)

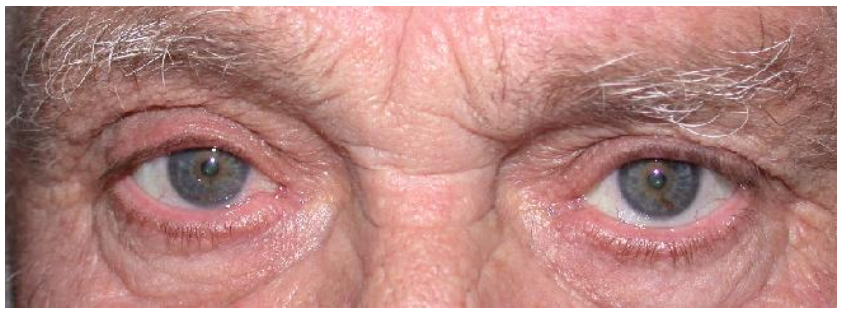
Classic problem: within-class variation (same person, different conditions) can exceed the between-class variation (different persons).

Changes in appearance over time (sometimes artificial and deliberate)

Paradox of facial phenotype and genotype

Facial appearance (phenotype) of everyone changes over time with age; but monozygotic twins (identical genotype) track each other as they age.

Therefore at any given point in time, they look more like each other than they look like themselves at either earlier or later periods in time

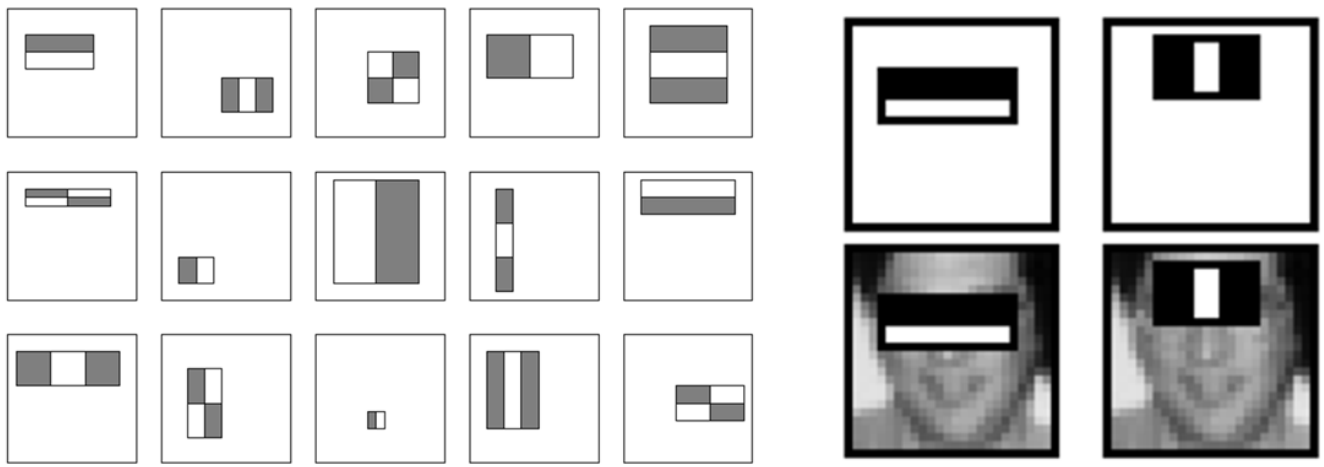
(Face detection, recognition, and interpretation, con't)

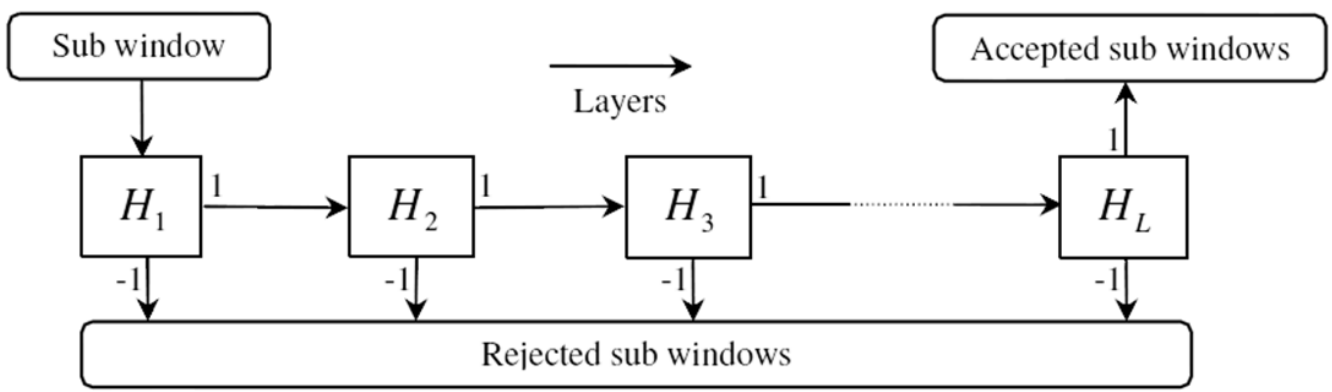
Detecting and recognising faces raises all the usual questions encountered in other domains of computer vision:

- What is the best representation to use for faces?
- Should this be treated as a 3D problem (object-based, volumetric), or a 2D problem (image appearance-based)?
- How can invariances to size (hence distance), location, pose, and illumination be achieved? (A given face should acquire a similar representation under such transformations, for matching purposes.)
- 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?
- What are the particular features that we can rely upon to distinguish among faces, and thus determine the identity of a given face?
- What is the best way to handle "integration of evidence", and incomplete information, and to make decisions under uncertainty?
- How can machine learning develop domain expertise, either about faces in general (e.g. pose transformations), or facial distinctions?

Viola-Jones face detection algorithm

Paradoxically, face detection is a harder problem than recognition, and performance rates of algorithms are poorer. (It seems paradoxical since detection precedes recognition; but recognition performance is measured only with images already containing faces.) The best known way to find faces is the cascade of classifiers developed by Viola and Jones (2004).





(Viola-Jones face detection algorithm, con't)

Key idea: build a strong classifier from a cascade of many weak classifiers – all of whom in succession must agree on the presence of a face

- A face (in frontal view) is presumed to have structures that should trigger various local "on-off" or "on-off-on" feature detectors
- A good choice for such feature detectors are 2D Haar wavelets (simple rectangular binary alternating patterns)
- There may be 2, 3, or 4 rectangular regions (each $+1$ or -1) forming feature detectors f_j, at differing scales, positions, and orientations
- Applying Haar wavelets to a local image region only involves adding and subtracting pixel values (no multiplications; hence very fast)
- A given weak classifier $h_j(x)$ consists of a feature f_j, a threshold θ_j and a polarity $p_j \in \pm 1$ (all determined in training) such that

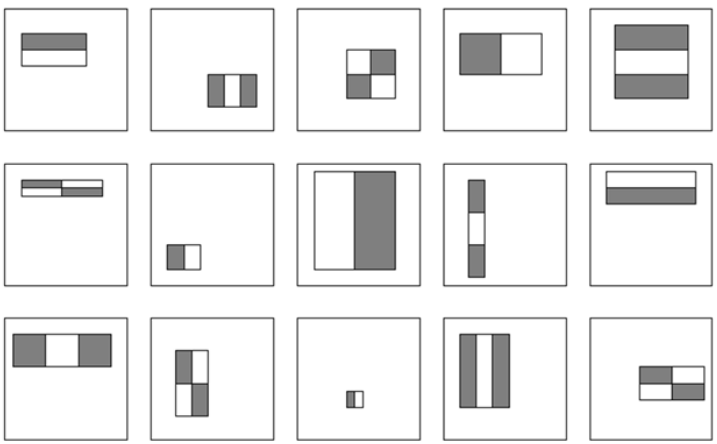
$$h_j(x) = \begin{cases} -p_j & \text{if } f_j < \theta_j \\ p_j & \text{otherwise} \end{cases}$$

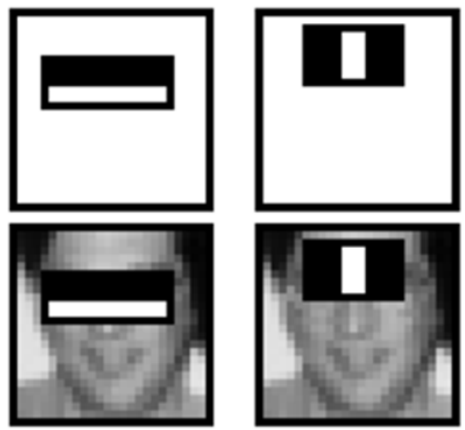
- A strong classifier $h(x)$ takes a linear combination of weak classifiers, using weights α_j learned in a training phase, and considers its sign:

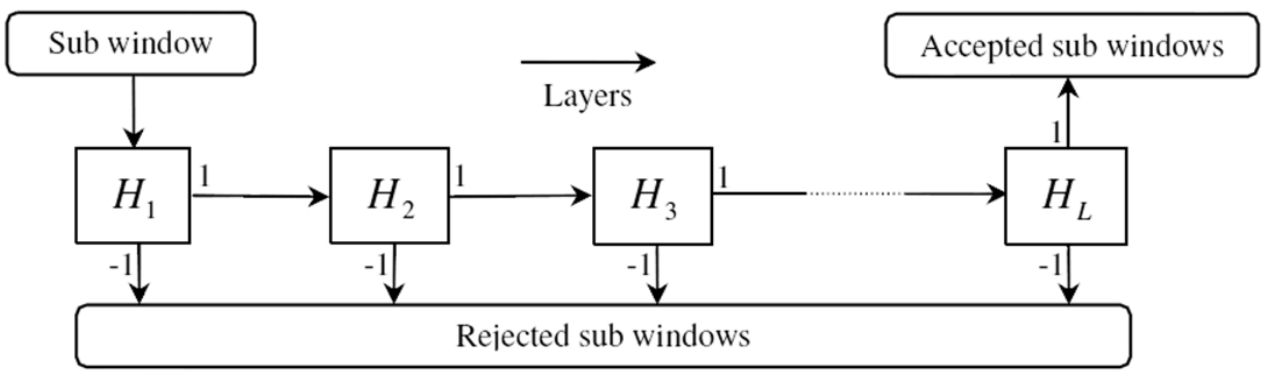
$$h(x) = \text{sign}(\sum_j \alpha_j h_j)$$

(Viola-Jones face detection algorithm, con't)

- At a given level of the cascade, a face is "provisionally deemed to have been detected" at a certain position if $h(x) > 0$
- Only those image regions accepted by a given layer of the cascade ($h(x) > 0$) are passed on to the next layer for further consideration
- A face detection cascade may have 30+ layers, yet the vast majority of candidate image regions will be rejected early in the cascade.







(Viola-Jones face detection algorithm, con't)

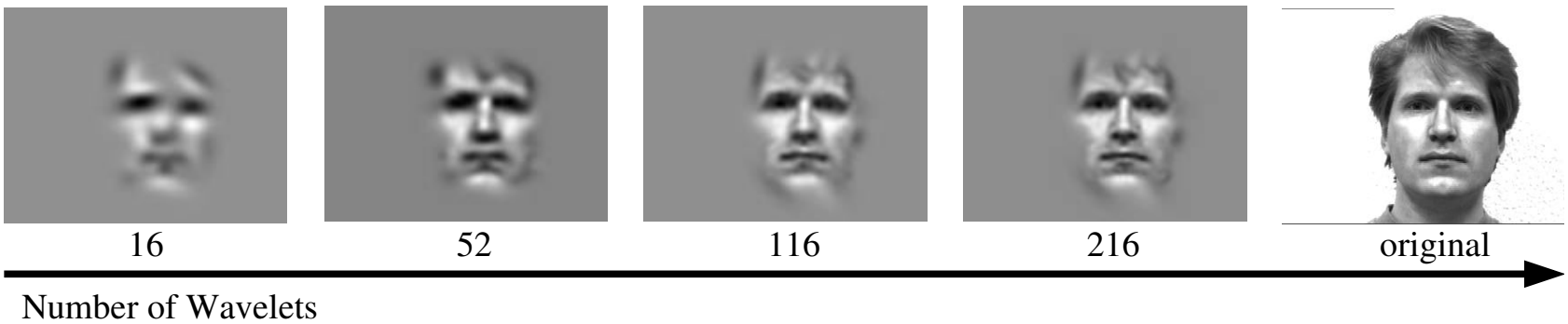
- Training uses the AdaBoost ("Adaptive Boosting") algorithm
- This supervised machine learning process adapts the weights α_j such that early cascade layers have very high true accept rates, say 99.8% (as *all* must detect a face; hence high false positive rates, say 68%)
- Later stages in the cascade, increasingly complex, are trained to be more discriminating and therefore have lower false positive rates
- More and more 2D Haar wavelet feature detectors are added to each layer and trained, until performance targets are met
- The cascade is evaluated at different scales and offsets across an image using a sliding window approach, to find any (frontal) faces
- With "true detection" probability d_i in the i^{th} layer of an N-layer cascade, the overall correct detection rate is: $D = \prod_{i=1}^{N} d_i$
- With "erroneous detection" probability e_i at the i^{th} layer, the overall false positive rate is $E = \prod_{i=1}^{N} e_i$ (as every layer must falsely detect)
- Example: if we want no false detections, with 10^5 image subregions so $E < 10^{-5}$, in a 30-layer cascade we train for $e_i = 10^{-5/30} \approx 0.68$ which shows why each layer can use such weak classifiers!
- Likewise, to achieve a decent overall detection rate of $D = 0.95$ requires $d_i = 0.95^{1/30} \approx .9983$ (very happy to call things "faces")

(Viola-Jones face detection algorithm, con't)

Performance on a local group photograph:

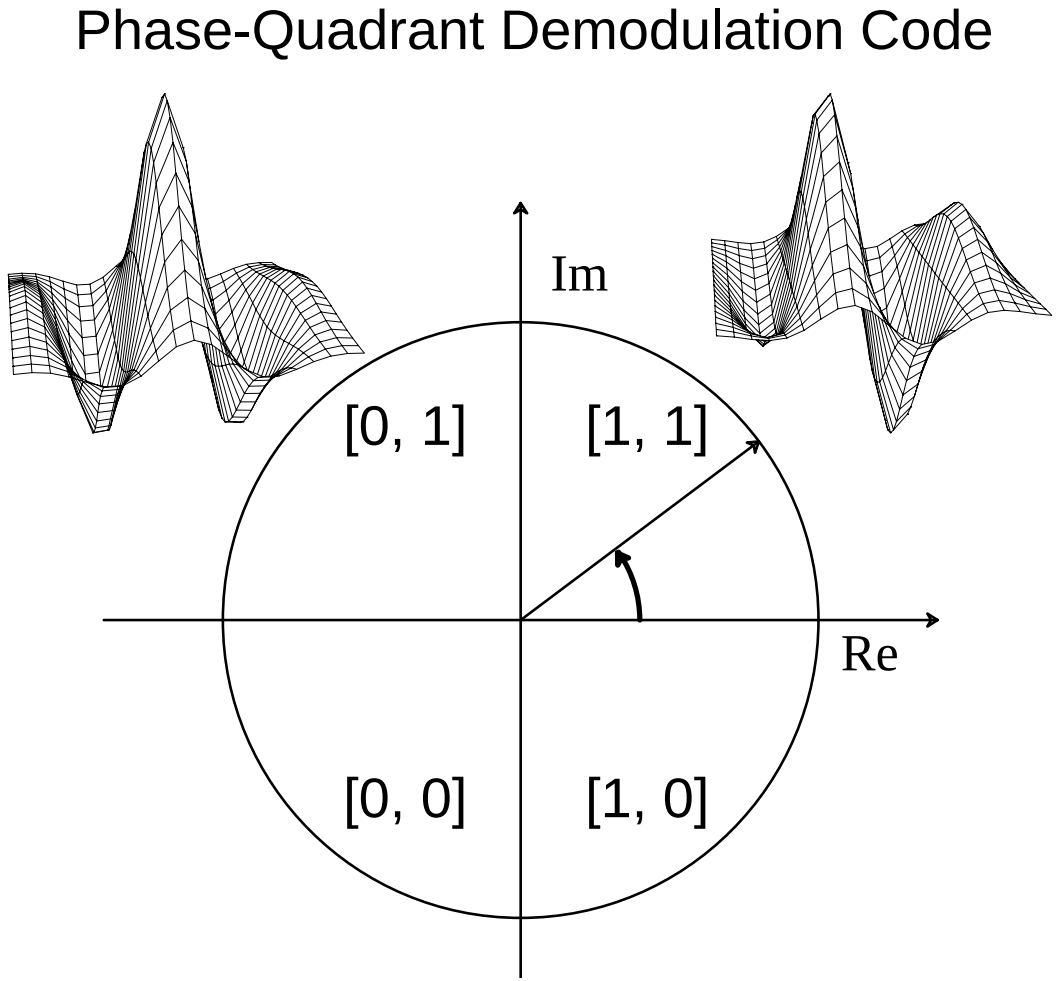
2D Appearance-based face recognition: Gabor wavelets

We saw that 2D Gabor wavelets can make remarkably compact codes for faces, among many other things. In this sequence, even using only about 100 Gabor wavelets, not only the presence of a face is obvious, but also its gender, rough age, pose, expression, and perhaps even identity:



- Gabor wavelets capture image structure as combined undulations
- Parameterisation: 2D positions, sizes, orientations, and phases
- Facial features like eyes, lips, and noses are represented with just a handful of wavelets, without requiring explicit *models* for such parts
- Can track changes of expression locally. Example: gaze = phase
- A deformable elastic graph made from such an encoding can preserve matching, while tolerating *some* changes in pose and expression

(2D Appearance-based face recognition: Gabor wavelets)



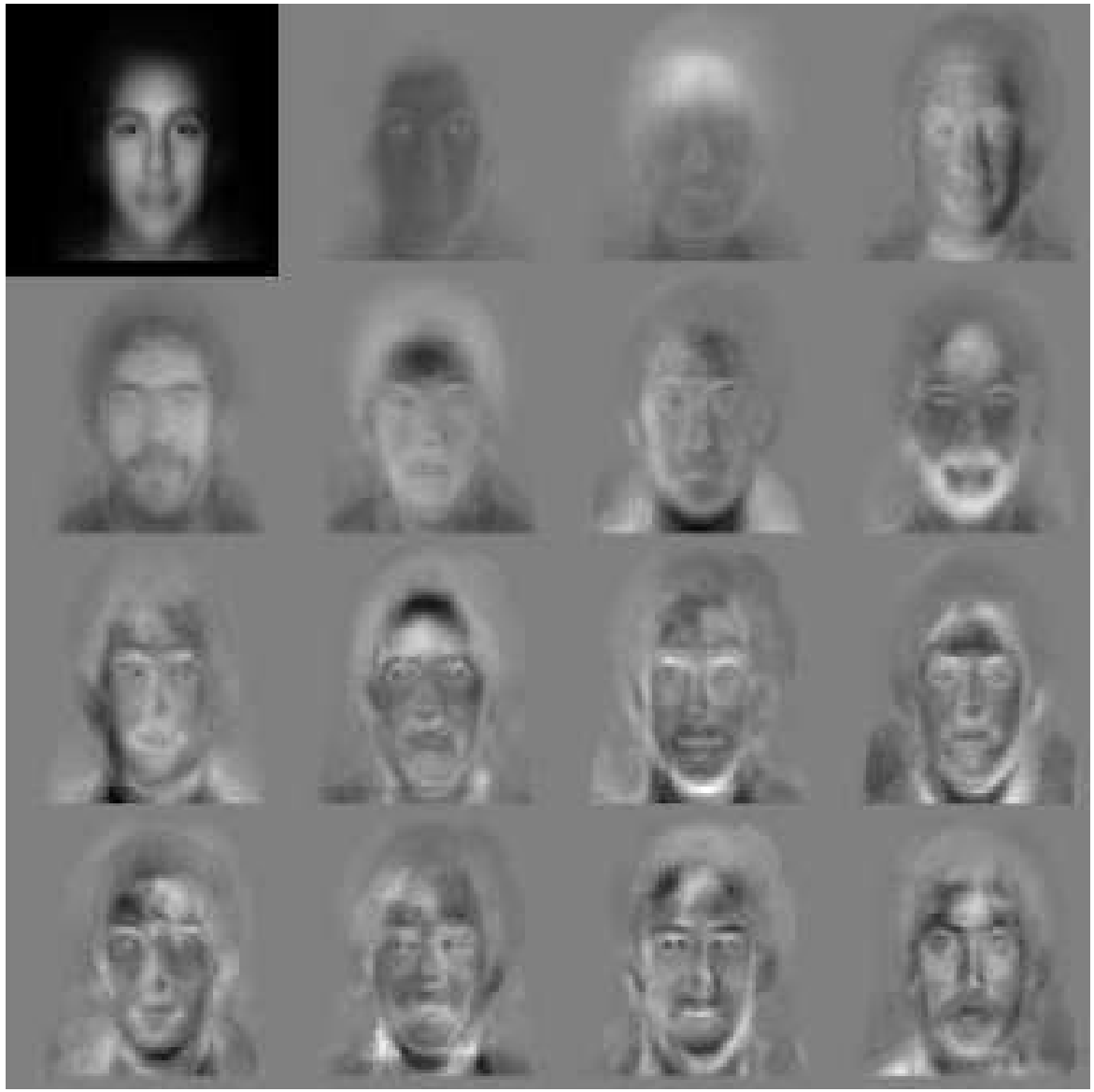
Computed feature vectors in a face code can be local 2D Gabor wavelet amplitude or phase information. Bits in the "face code" are set by the quadrant in which the phasor lies, for each aspect of facial structure.

2D Appearance-based face recognition: "eigenfaces"

An elegant method for 2D appearance-based face recognition combines Principal Components Analysis (PCA) with machine learning and algebra, to compute a linear basis (like the Fourier basis) for representing any face as a combination of empirical eigenfunctions, called eigenfaces.

- A database of face images (at least 10,000) that are pre-normalised for size, position, and frontal pose is "decomposed" into its Principal Components of statistical variation, as a sequence of orthonormal eigenfunctions whose eigenvalues are in descending order
- This is a classical framework of linear algebra, associated also with the names Karhunen-Loève Transform, or the Hotelling Transform, or Dimensionality Reduction and subspace projection
- Optimised for truncation: finding the best possible (most accurate) representation of data using any specified finite number of terms
- Having extracted from a face gallery the (say) 20 most important eigenfaces of variation (in sequence of descending significance), any given presenting face is projected onto these, by inner product
- The resulting (say) 20 coefficients then constitute a very compact code for representing, and recognising, the presenting face
- 15 such representational eigenfaces are shown in the next slide

(2D Appearance-based face recognition: "eigenfaces")



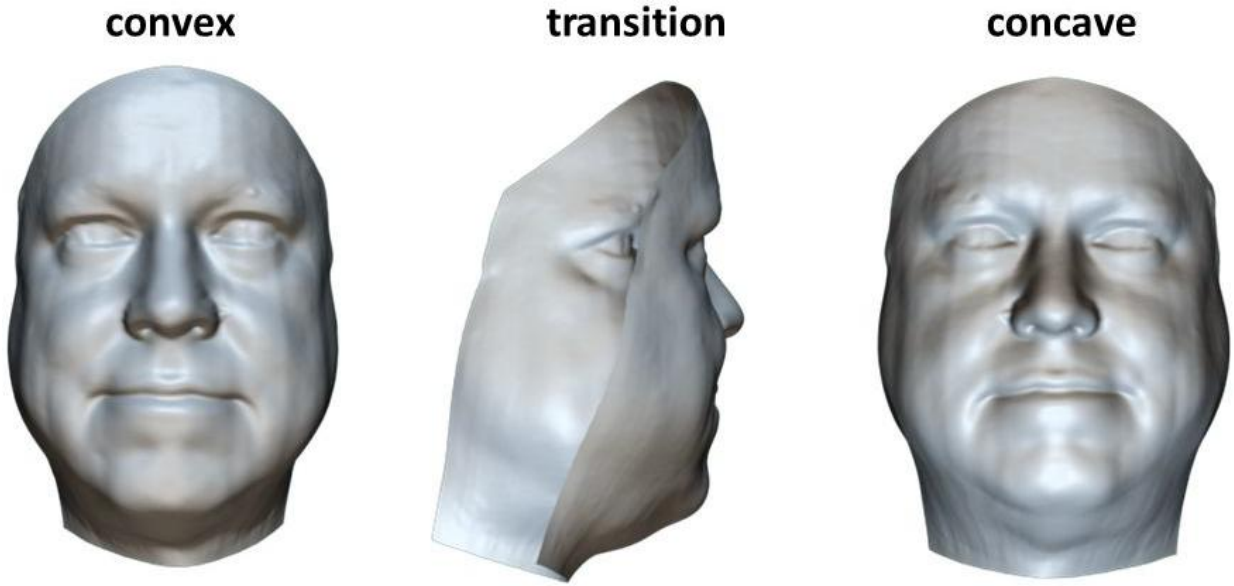
The top left face is a particular linear combination of the eigenfaces

(2D Appearance-based face recognition: "eigenfaces")

- Performance is often in the range of 90% to 95% accuracy
- Databases can be searched very rapidly, as each face is represented by a very compact feature vector of only about 20 numbers
- A major limitation is that significant (early, low-order) eigenfaces emerging from the statistical analysis arise just from normalisation errors of size (head outlines), or variations in illumination angle
- Like other 2D representations for faces, the desired invariances for transformations of size (distance), illumination, and pose are lacking
- Both the Viola-Jones face detection algorithm, and these 2D appearance-based face recognition algorithms, sometimes deploy "brute force" solutions (say at airport Passport control) such as acquiring images from a large (3×3) or (4×4) array of cameras for different pose angles, each allowing some range of angles

Our irresistible Bayesian priors when processing faces

The rotating hollow mask illusion illustrates the overwhelming strength of our "prior" belief that faces are mostly convex, not concave. As the mask rotates into concave presentation, we cannot resist the false perception that it has reversed its direction of rotation.



Video demo with texture mapping: www.youtube.com/watch?v=01LMFFpAWYM , or without: www.youtube.com/watch?v=sKa0eaKsdA0

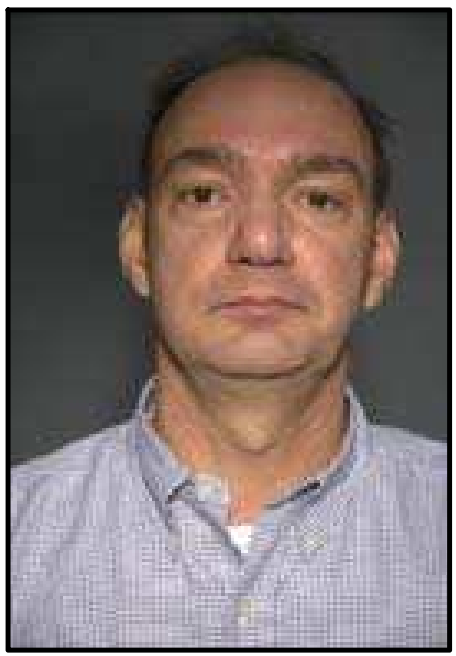
Once the rotation angle makes the visible area increase again, the only way to preserve convexity is to see the direction of rotation as reversing.

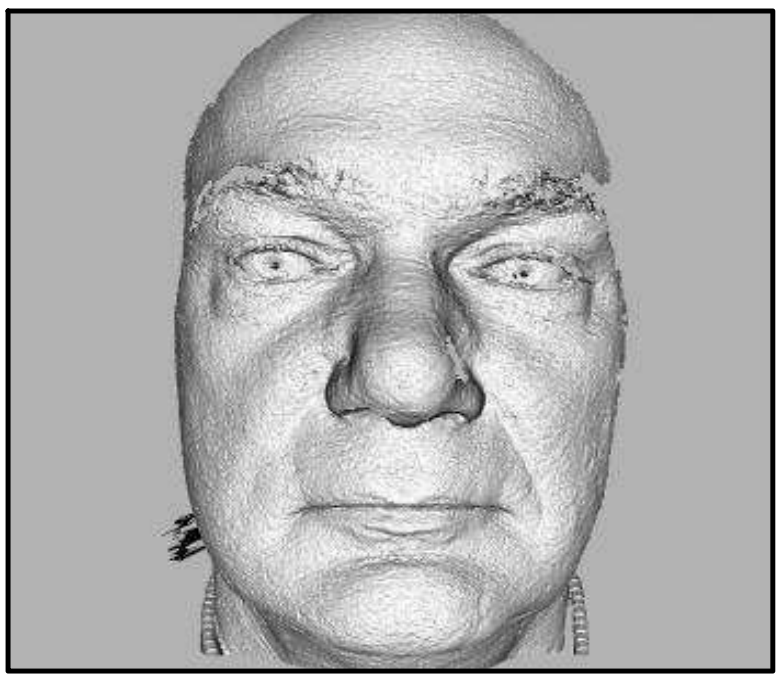
Is this a "feature" or a "bug"? Should we try to design in such illusions into computer vision? Or just hope they do arise, as a sign of success?

Three-dimensional approaches to face recognition

Face recognition algorithms now aim to model faces as three-dimensional objects, even as dynamic objects, in order to achieve invariances for pose, size (distance), and illumination geometry. Performing face recognition in object-based (volumetric) terms, rather than appearance-based terms, unites vision with model-building and graphics.

To construct a 3D representation of a face, it is necessary to extract both a shape model (below right), and a texture model (below left). The term "texture" here encompasses albedo, colouration, and 2D surface details.





(Three-dimensional approaches to face recognition)

Extracting the 3D shape model can be done by various means:

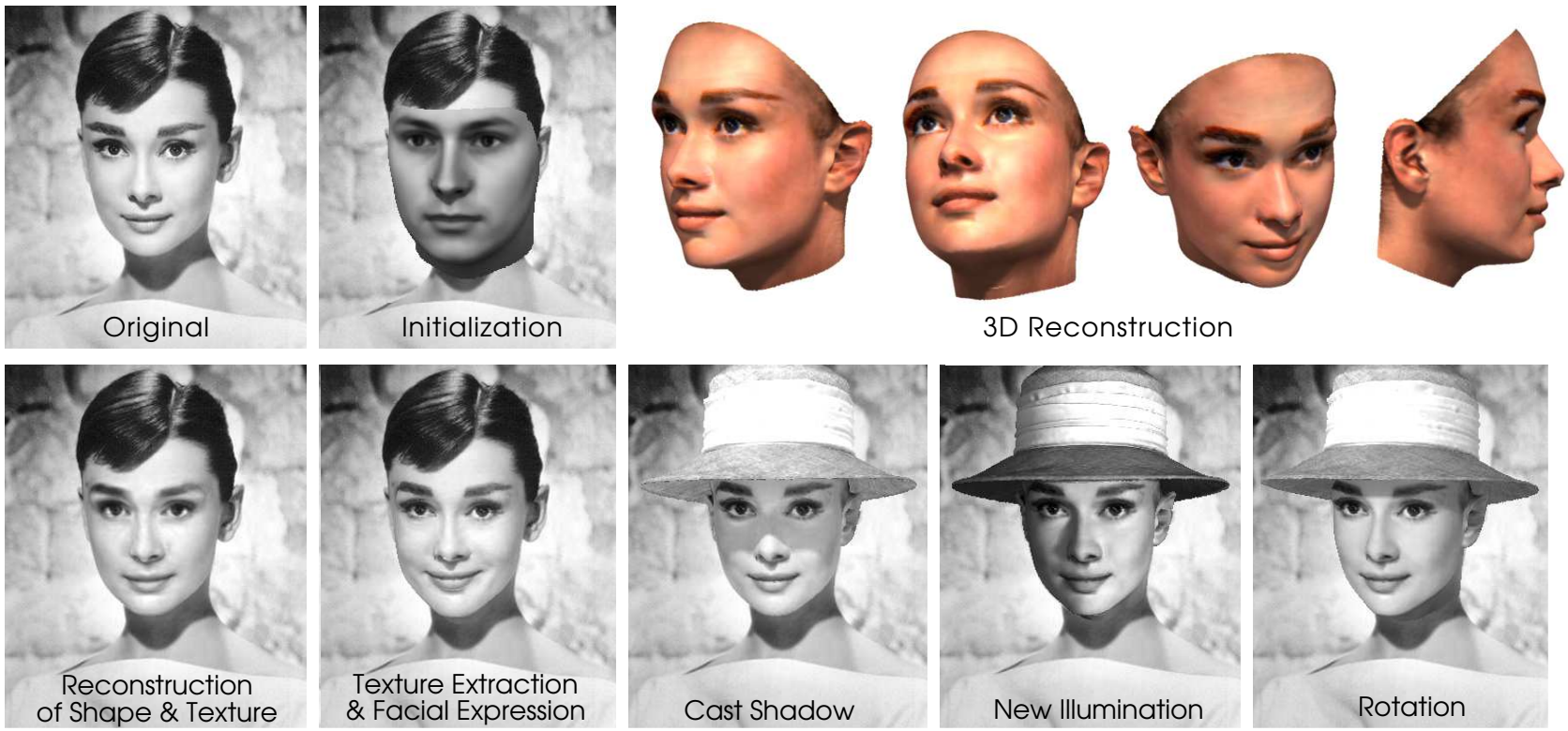
- laser range-finding, even down to millimetre resolution
- calibrated stereo cameras
- projection of structured IR light (grid patterns whose distortions reveal shape, as with Kinect)
- extrapolation from multiple images taken from different angles

The size of the resulting 3D data structure can be in the gigabyte range, and significant time can be required for the computation.

Since the texture model is linked to coordinates on the shape model, it is possible to "project" the texture (tone, colour, features) onto the shape, and thereby to generate predictive models of the face in different poses.

Clearly sensors play an important role here for extracting shape models, but it is also possible to do this even from just a single photograph if sufficiently strong Bayesian priors are also marshalled, assuming an illumination geometry and some universal aspects of head and face shape.

(Three-dimensional approaches to face recognition)



An impressive demo of using a single 2D photograph (top left) to morph a 3D face model after manual initialisation, building a 3D representation of the face that can be manipulated for differing pose angles, illumination geometries, and even expressions, can be seen here:

http://www.youtube.com/watch?v=nice6NYb_WA

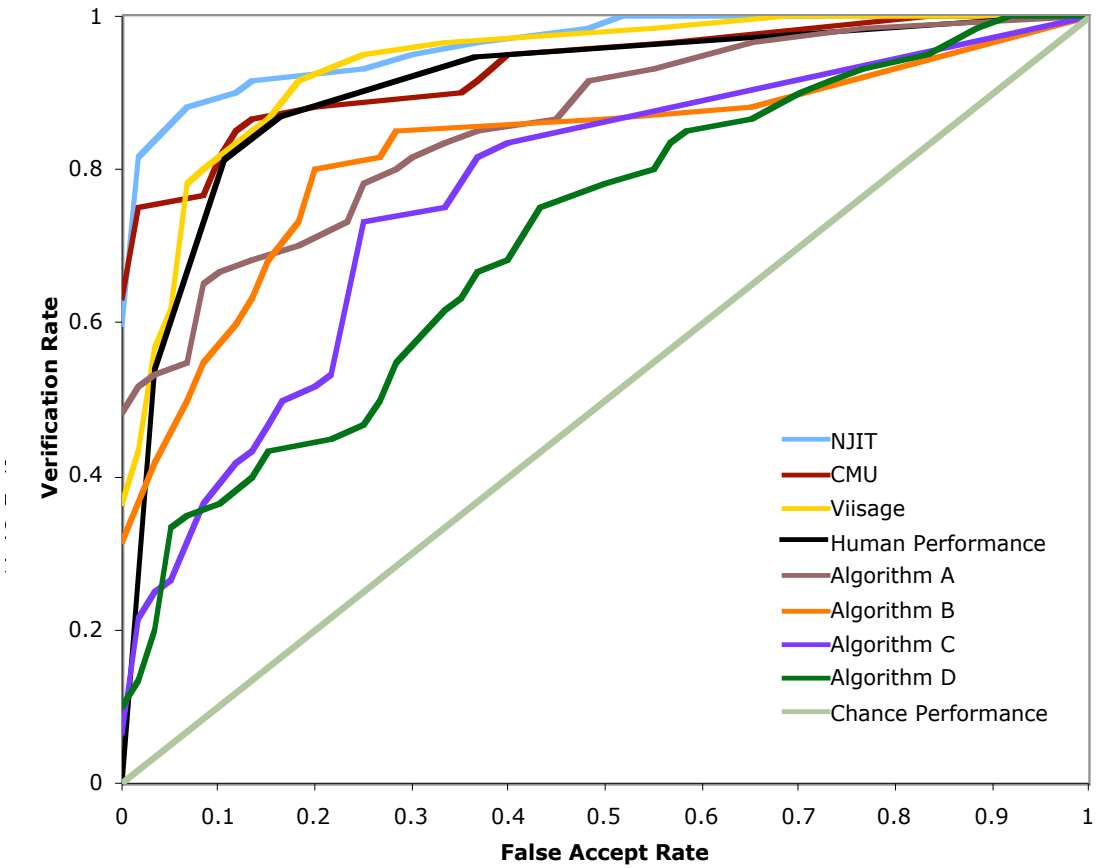
(Three-dimensional approaches to face recognition)

Description from the Blanz and Vetter paper,
Face Recognition Based on Fitting a 3D Morphable Model:

"...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."

Face algorithms compared with human performance

The US National Institute for Standards and Technology (NIST) runs periodic competitions for face recognition algorithms, over a wide range of conditions. Uncontrolled illumination and pose remain challenging. In a recent test, three algorithms had ROC curves above (better than) human performance at non-familiar face recognition (the black curve). But human performance remains (as of 2018) better on familiar faces.



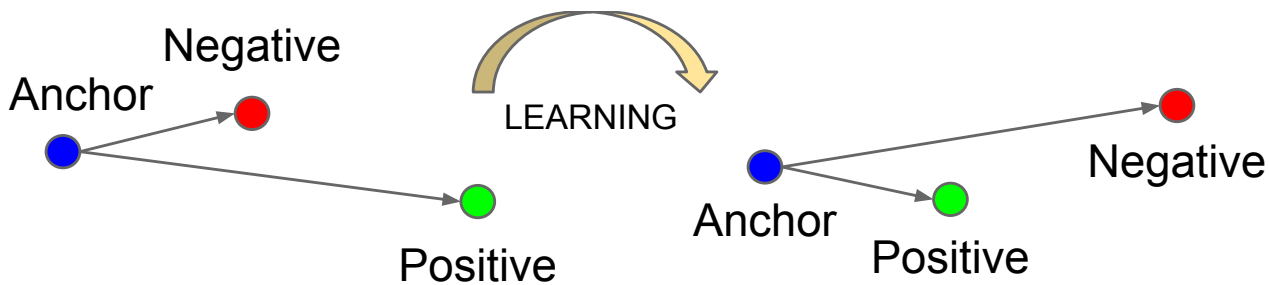
Major breakthrough with CNNs: deep-learning 'FaceNet'

Machine learning approaches focused on scale ("Big Data") are having a profound impact in Computer Vision. In 2015 Google demonstrated large reductions in face recognition error rates (by 30%) on two very difficult databases: YouTube Faces (95%), and Labeled Faces in the Wild (LFW) database (99.63%), which remain as accuracy records. But when tested on larger ("MegaFace") datasets, accuracy fell to about 75%.

(Major breakthrough with CNNs: deep-learning 'FaceNet')

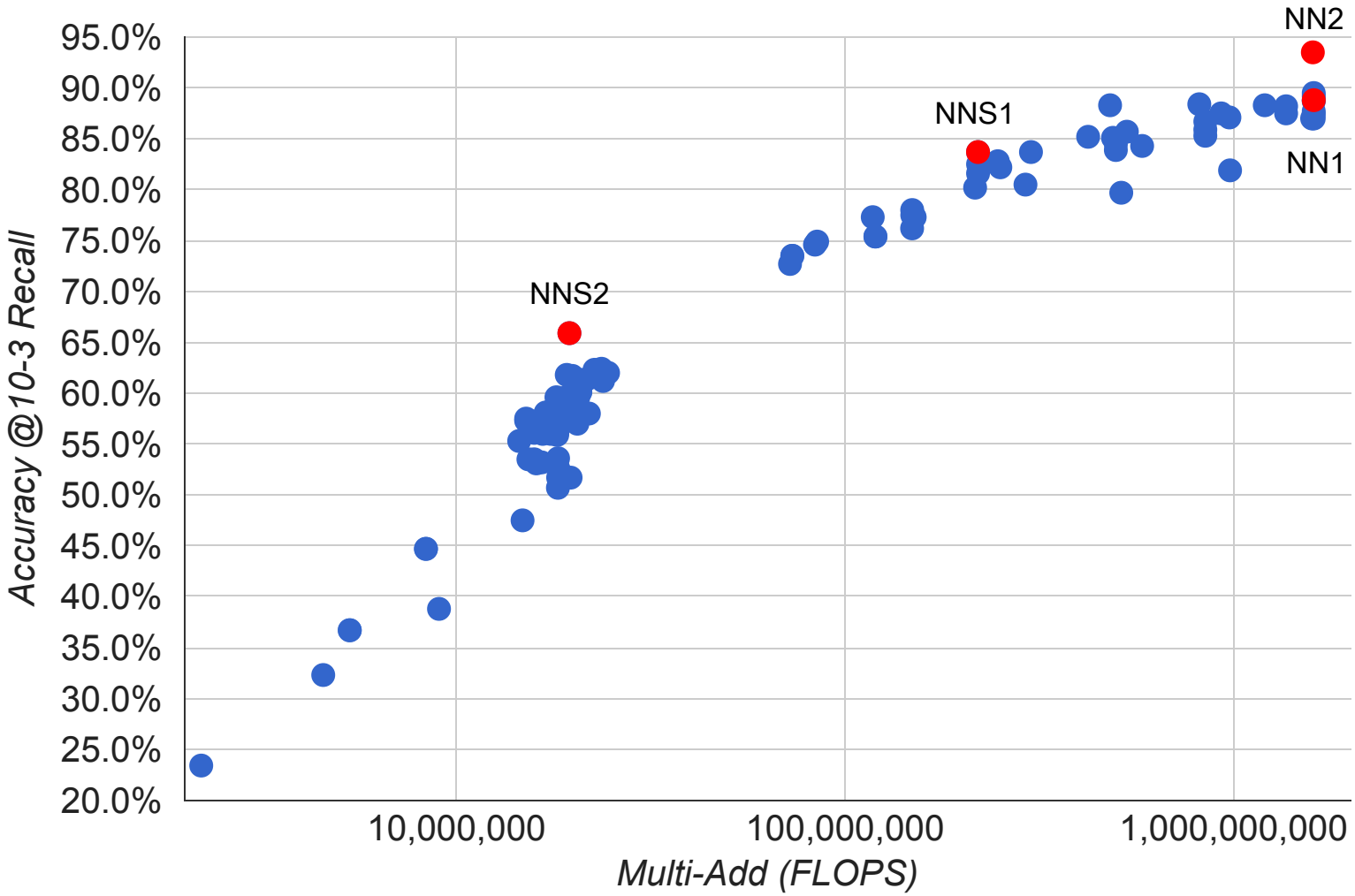
- Convolutional Neural Net with 22 layers and 140 million parameters
- Big dataset: trained on 200 million face images, 8 million identities
- 2,000 hours training (clusters); about 1.6 billion FLOPS per image
- Euclidean distance metric (L2 norm) on embeddings $f(x_i)$ learned for cropped, but not pre-segmented, images x_i using back-propagation
- Used triplets of images, one pair being from the same person, so that both the **p**ositive (same face) and **n**egative (different person) features were learned by minimising a loss function L:

$$L = \sum_i \left[\| f(x_i^a) - f(x_i^p) \|^2 - \| f(x_i^a) - f(x_i^n) \|^2 \right]$$



- The embeddings create a compact (128 byte) code for each face
- Simple threshold on Euclidean distances among these embeddings then gives decisions of "same" vs "different" person

(Major breakthrough with CNNs: deep-learning 'FaceNet')



Different variants of the Convolutional Neural Net and model sizes were generated and run, revealing the trade-off between FLOPS and accuracy for a particular point on the ROC curve (False Accept Rate = 0.001)

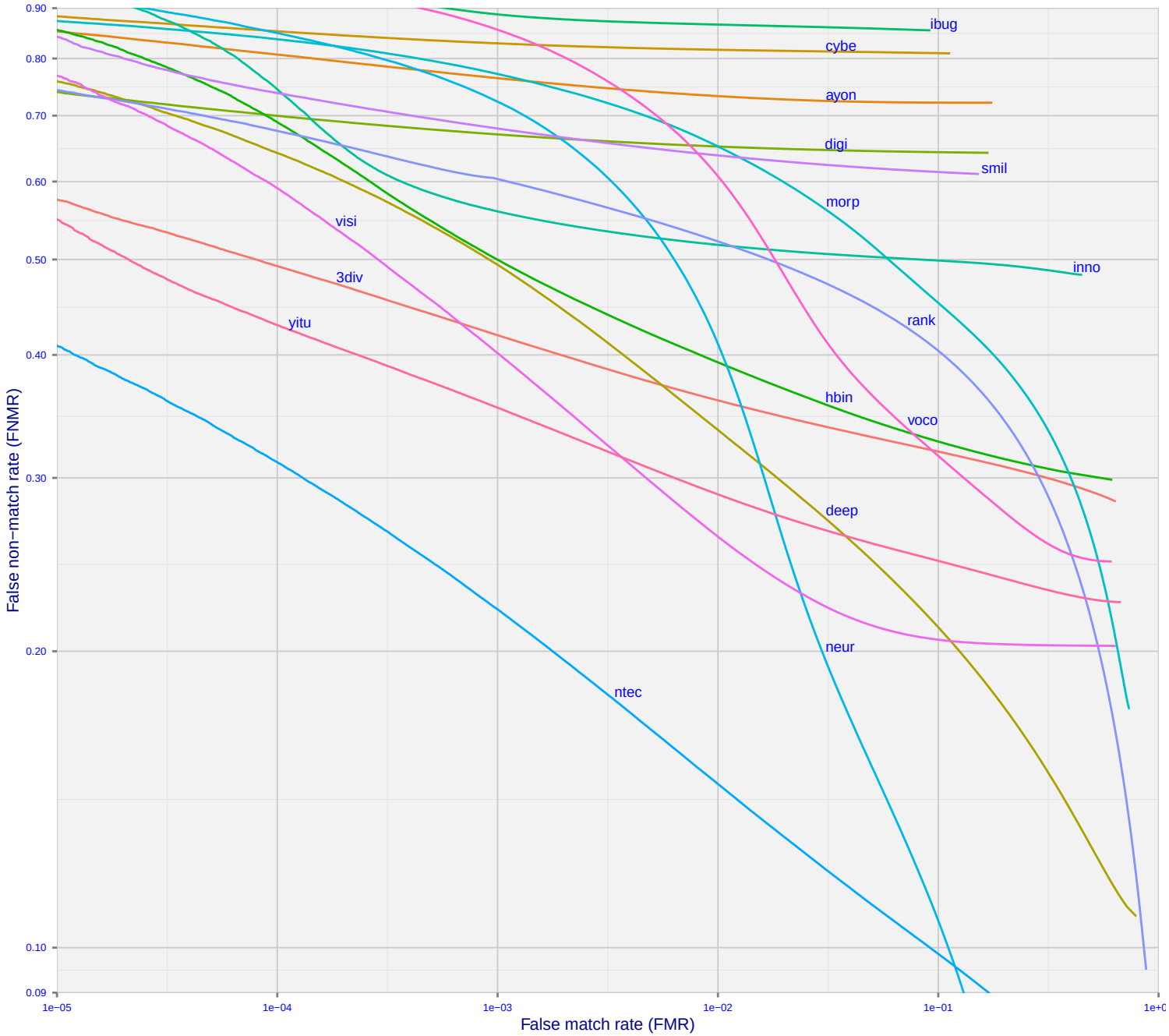
2017 IARPA (US DoD) Face Recognition Competition

- Major NIST test commissioned by several US intelligence agencies
- Search a gallery of "cooperative portrait photos" of $\sim$ 700K faces...
- ... using non-ideal probes: face photos without quality constraints:
  - persons unaware of, and not cooperating with, the image acquisition
  - variations in head pose, facial expression, illumination, occlusion
  - reduced image resolution (e.g. photos taken from a distance)
- Face image databases were"web-scraped, ground-truthed"
- Competitors: 16 commercial and academic entries, all trained on vast databases using advanced machine learning algorithms (CNNs) to learn invariances for pose, expression, illumination, occlusion
- Metrics and benchmarks of the competition:
  - 1-to-1 verification with lowest False non-Match Rate (FnMR) when the False Match Rate (FMR) threshold is set to FMR $= 0.001$
  - identification accuracy: lowest FnMR when FMR $= 0.001$ while searching a gallery of $\sim$ 700,000 images
  - identification speed: fastest search of $\sim$ 700,000 identities while FnMR remains good

Highlights of 2017 Face Recognition Competition results

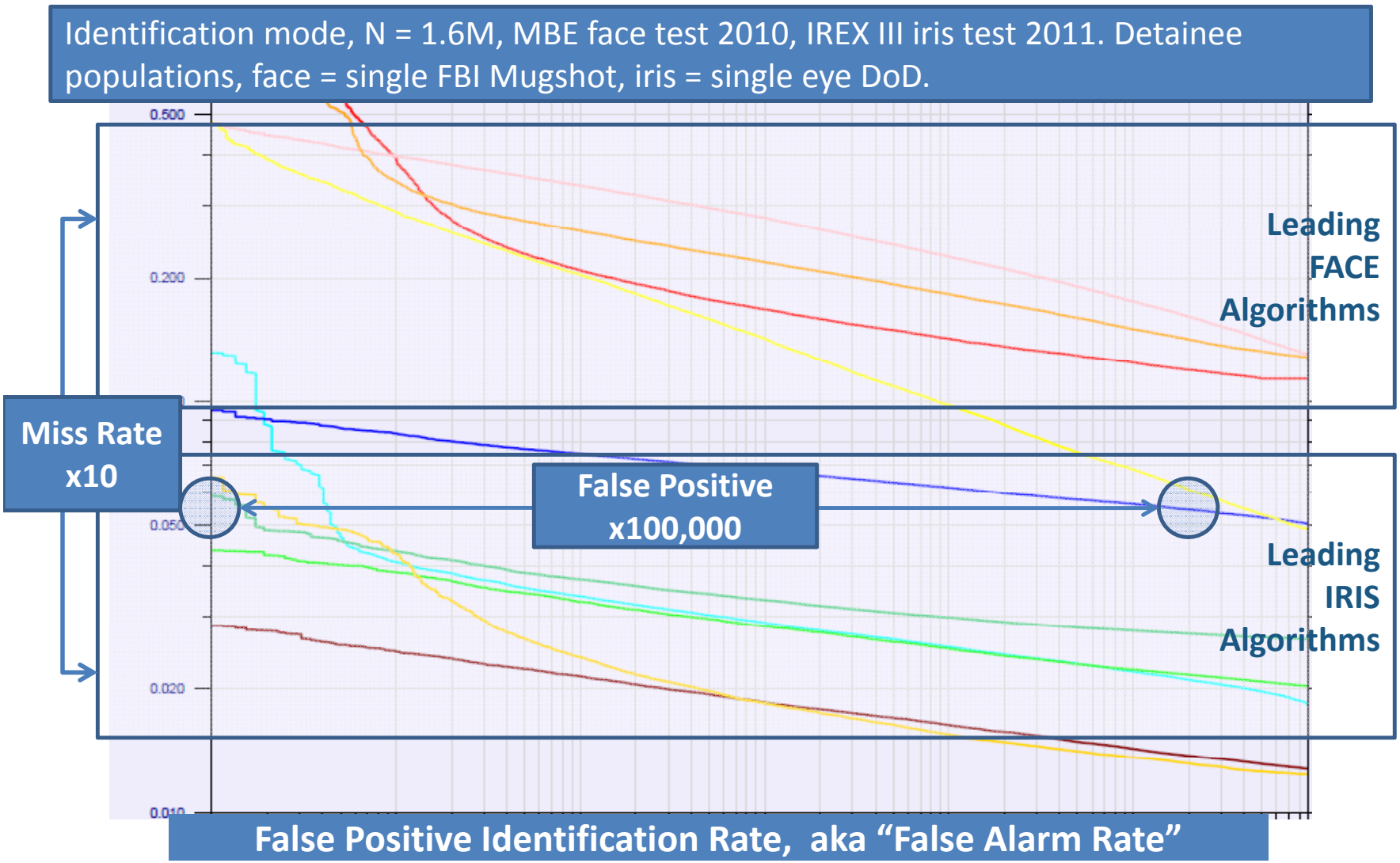
- identification accuracy: FnMR = 0.204 achieved at FMR = 0.001
- successful indexing instead of exhaustive search through a gallery: matches retrieved from 700K image gallery in just 0.6 milliseconds! (One process, running on a single core of a c. 2016 server-class CPU)
- sub-linear scaling of search time: a 30-fold increase in gallery size incurs only a 3-fold increase in search duration, for fastest entry
- (obviously humans don't perform sequential searches through a memory bank of previously seen faces in order to recognise a face)
- but building the fast-search index on 700,000 images takes 11 hours

Decision Error Trade-off curves, 2017 face recognition



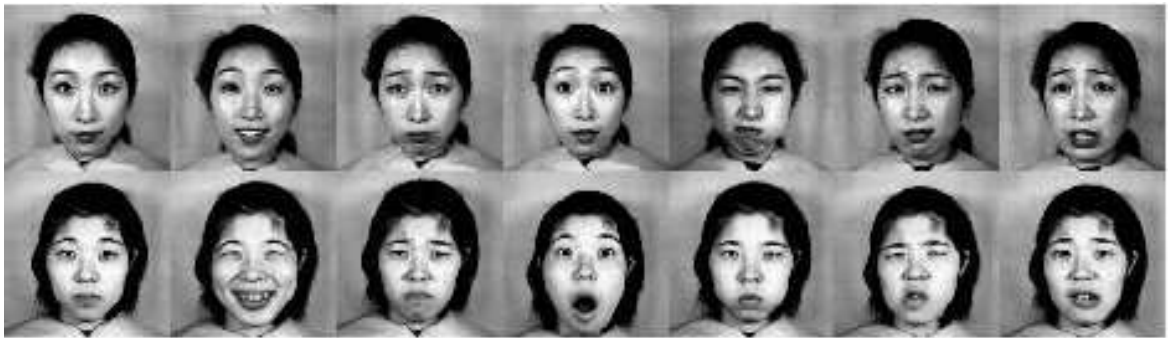
NIST also made DET curves for face versus iris recognition

- Because of much greater entropy, IrisCode FMR was 100,000 x lower
- IrisCode FnMR was also 10 x lower than face recognition algorithms

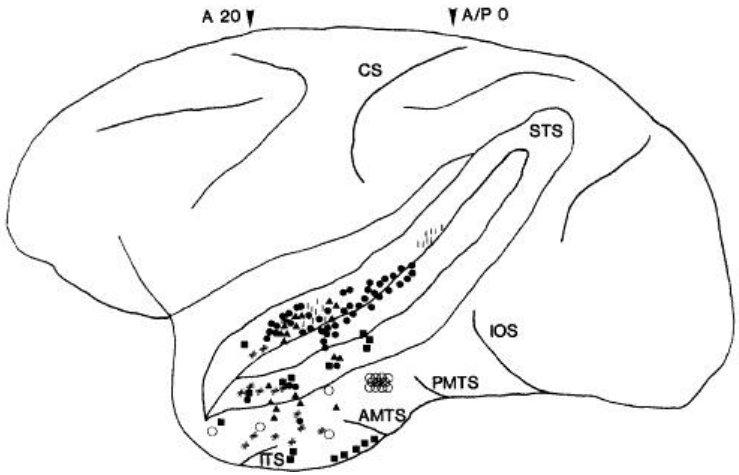


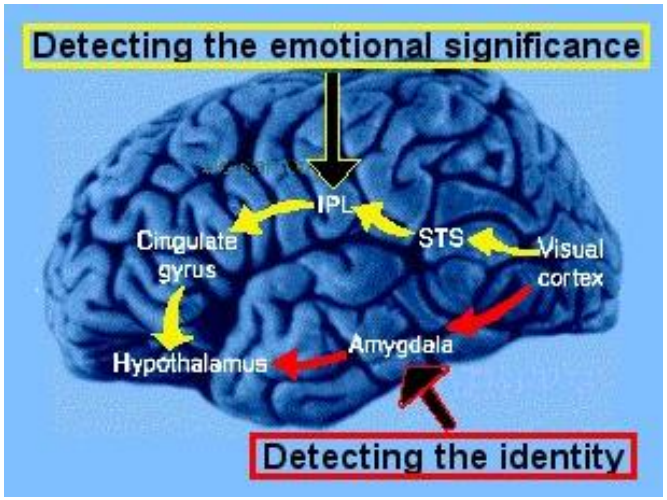
Affective computing: interpreting facial emotion

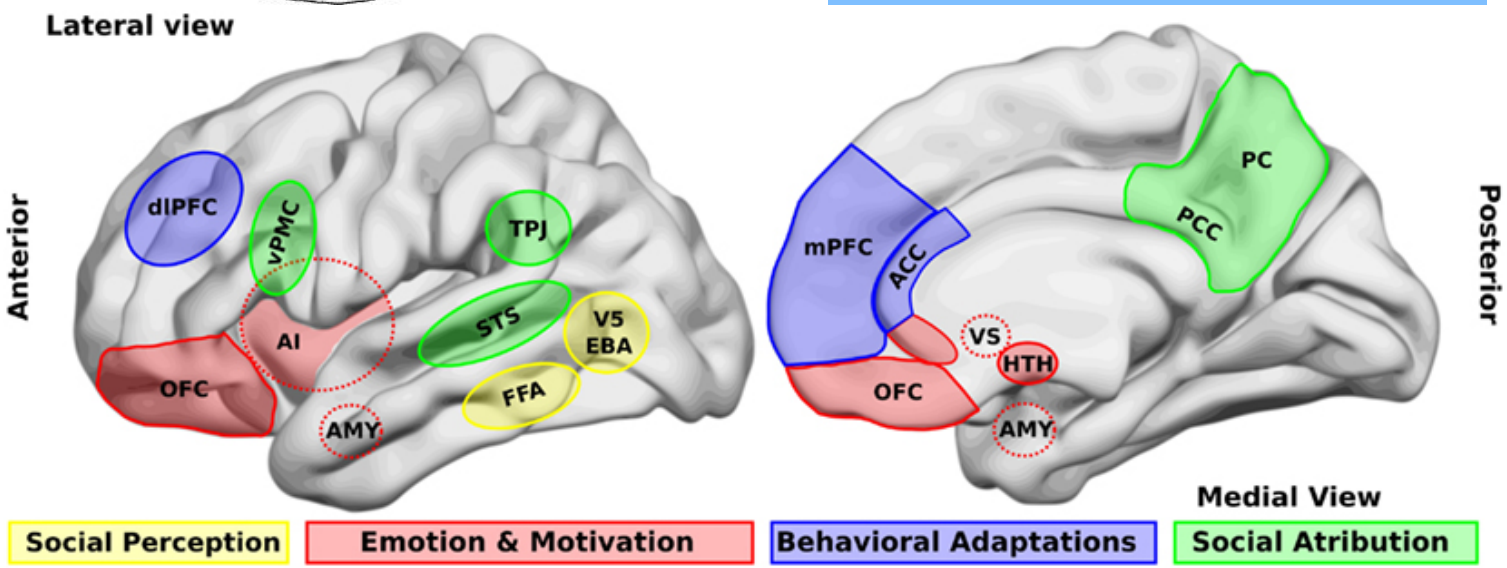
Humans use their faces as visually expressive organs, cross-culturally



Many areas of the human brain are concerned with recognising and interpreting faces, and social computation is believed to have been the primary computational load in the evolution of our brains, because of its role in reproductive success. (Genes survive; not individuals.)







Affective computing: classifying identity *and* emotion

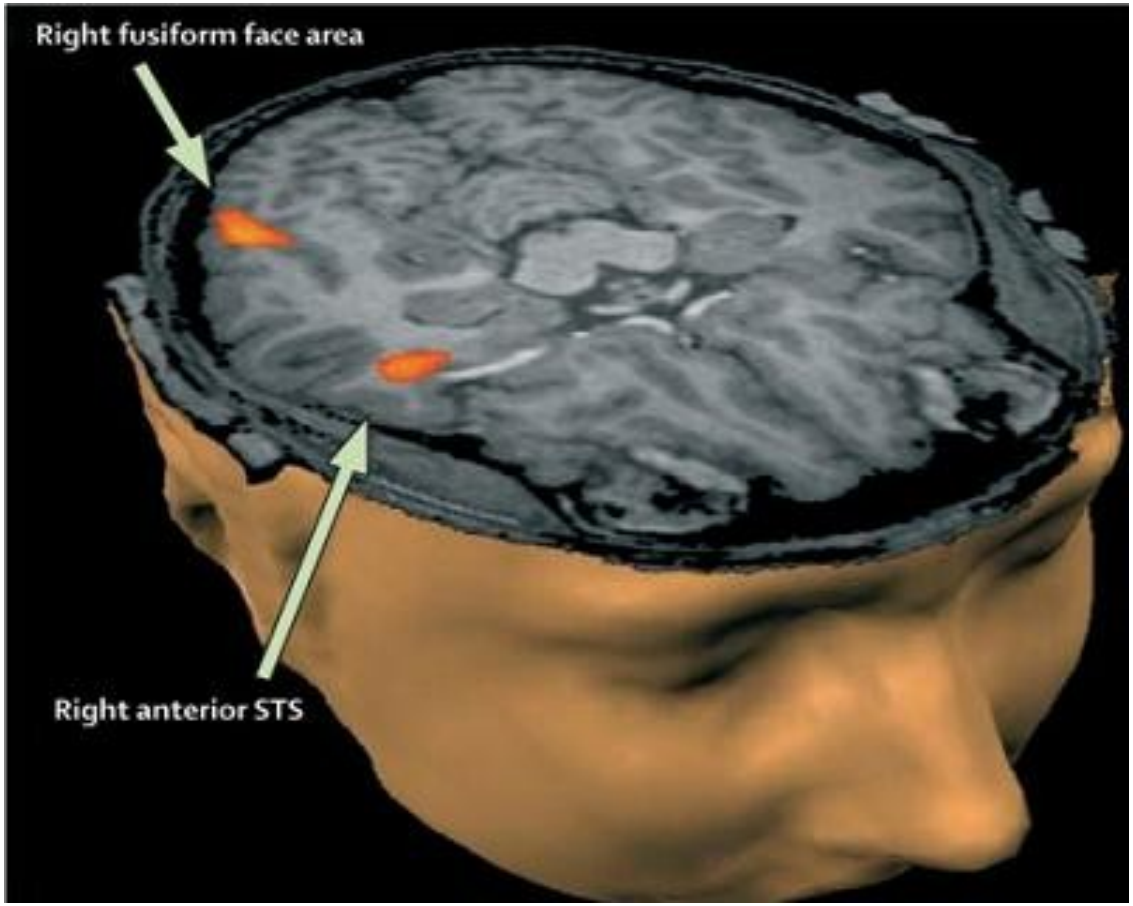
Target stimulus

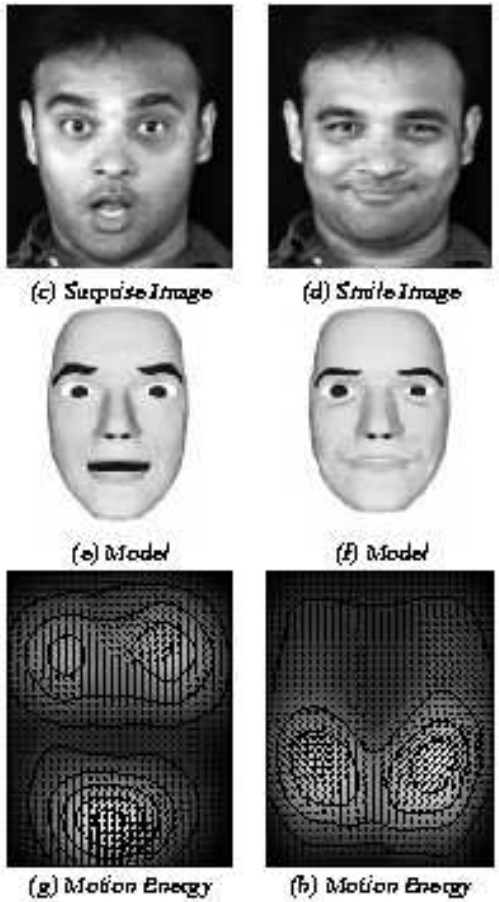
Same identity / different emotion

Same emotion / different identity

Different identity / different emotion

(Affective computing: interpreting facial emotion)





MRI scanning has revealed much about brain areas that interpret facial expressions. Affective computing aims to classify visual emotions as articulated sequences using Hidden Markov Models of their generation. Mapping the visible data to action sequences of the facial musculature becomes a generative classifier of emotions.

Facial Action Coding System (FACS)

- FACS is a taxonomy of human facial expressions
- It specifies 32 atomic facial muscle actions called Action Units (AU) executed by the seven articulating pairs of facial muscles
- Classically humans display six basic emotions: anger, fear, disgust, happiness, sadness, surprise, through prototypical expressions
- Ethnographic studies suggest these are cross-cultural universals
- FACS also specifies 14 Action Descriptors (ADs): e.g. head pose, gaze direction, thrust of the jaw, mandibular actions
- Message judgement decodes meanings of these objective encodings
- Analysis is subtle: e.g. distinguishing polite versus amused smiles
- Promising applications:
  - understanding human mental state; attributing feelings
  - detecting intentions, detecting deceit
  - affective computing; understanding nonverbal communication
  - building emotion interfaces
  - prediction of human behaviour

(Facial Action Coding System FACS, con't)

- Pre-processing: face detection; normalisation; facial point tracking
- Feature extraction, e.g. 2D Gabor features (usually 8 orientations, and 5 to 9 frequencies) are powerful to detect facial landmarks, and for representing wrinkling and bulging actions
- Appearance-based, geometry, motion, or hybrid approaches
- Spatio-temporal appearance features in video, versus static frames
- AU temporal segmentation, classification, and intensity estimation
- Coding the dynamic evolution between facial displays in videos
- Generative models (used with active appearance models) aim to infer emotional state by modeling the muscular actions that generated it
- Discriminative methods fit deformable models and train a classifier
- Hidden Markov Models trained on articulated facial expressions

Facial Expression and Analysis: algorithm tests

- Just like the Face Recognition Competitions, there are FERA: Facial Expression and Analysis Challenges (2011, 2015, 2017)
- Metrics used: Occurrence detection, and Intensity estimation
- Facial action detection measured with varying head pose
- Disappointing results so far (2017), compared to face recognition:
  - Occurence detection accuracy: ~ 0.57
  - Intensity estimation accuracy: ~ 0.44
- Limitations: training sets were often non-spontaneous expressions; small datasets; large subject differences; environmental influences
- Building database 'ground truths': more than 100 hours of training is required to become a human expert FACS coder
- Manual scoring: each minute of video requires about an hour
- Facial AU analysis remains an underdeveloped field with many open issues but enormous potential for more fluid HCI interfaces