Computer Vision

Professor John Daugman

University of Cambridge

Computer Science Tripos, Part II
Lent Term 2015/16

Outline of Lectures

1. **Overview. Goals of computer vision; why they are so difficult.**
2. **Image sensing, pixel arrays, CCD cameras. Image coding.**
3. **Biological visual mechanisms, from retina to visual cortex.**
4. **Mathematical operations for extracting structure from images.**
5. **Edge detection operators; gradients; zero-crossings of Laplacian.**
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.

Recommended books and online resources

- Forsyth, D.A. & Ponce, J. (2003). *Computer Vision: A Modern Approach*.
- Shapiro, L. & Stockman, G. (2001). *Computer Vision*. Prentice Hall.
- Duda, R.O., Hart, P.E., & Stork, D.G. (2001) *Pattern Classification* (2nd Ed).

- CVonline: "Evolving, Distributed, Non-Proprietary, On-Line Compendium of Computer Vision" (Univ. of Edinburgh; updated Aug. 2015; includes many Wikipedia links): http://homepages.inf.ed.ac.uk/rbf/CVonline/
- Matlab Functions for Computer Vision and Image Processing (updated July 2015): http://www.peterkovesi.com/matlabfns/index.html
- Annotated Computer Vision Bibliography (updated 1 Jan. 2016): http://iris.usc.edu/Vision-Notes/bibliography/contents.html
- 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).
- A collection of **Practical Exercises** for this course developed by C Richardt, T Baltrusaitis, and L Swirski is provided here: http://www.cl.cam.ac.uk/~ls426/computervision/

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

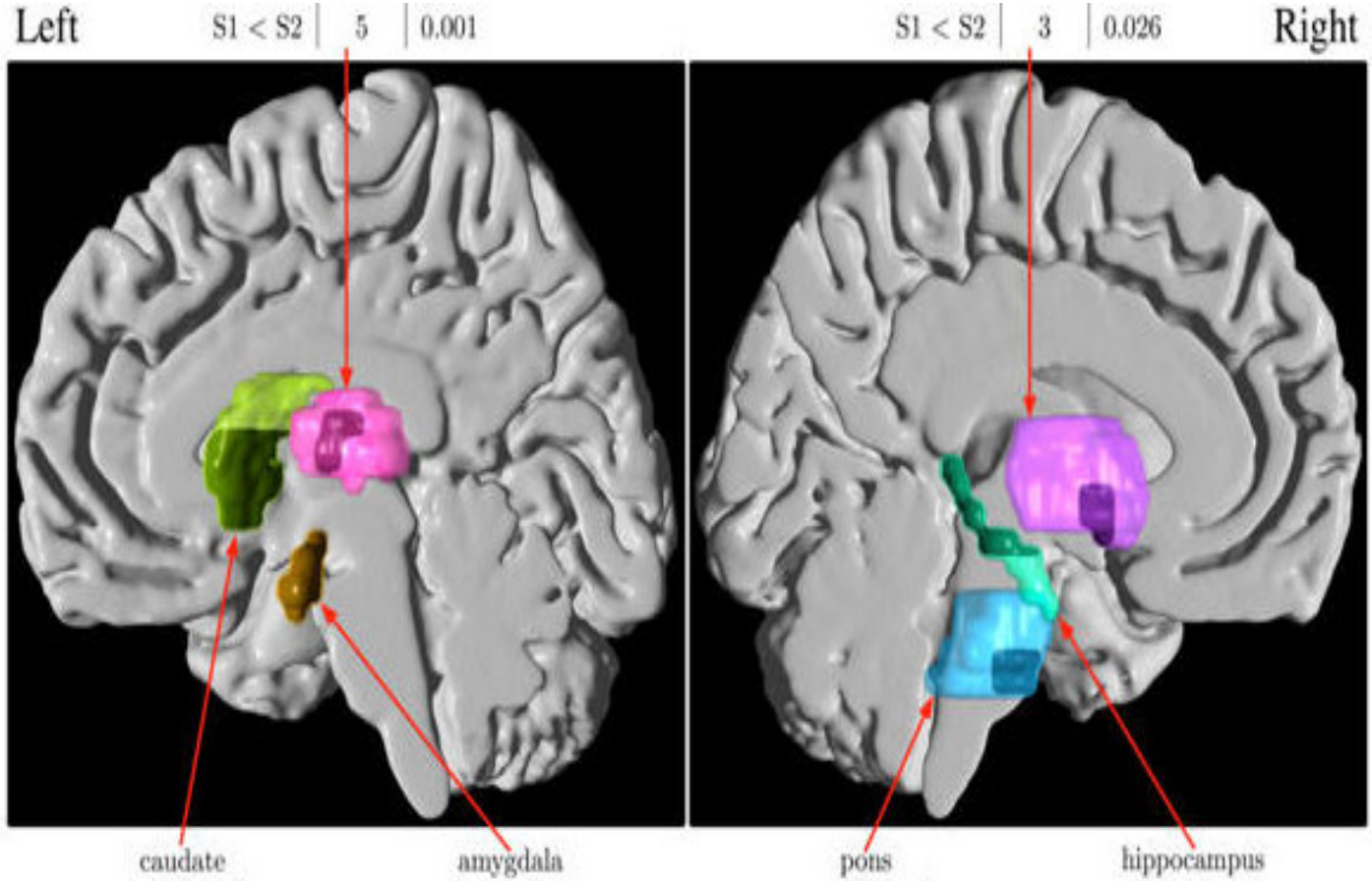
- object and pattern recognition; 3D scene reconstruction from images
- biometric-based visual determination of personal identity
- image search and content-based image retrieval; scene understanding

(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)

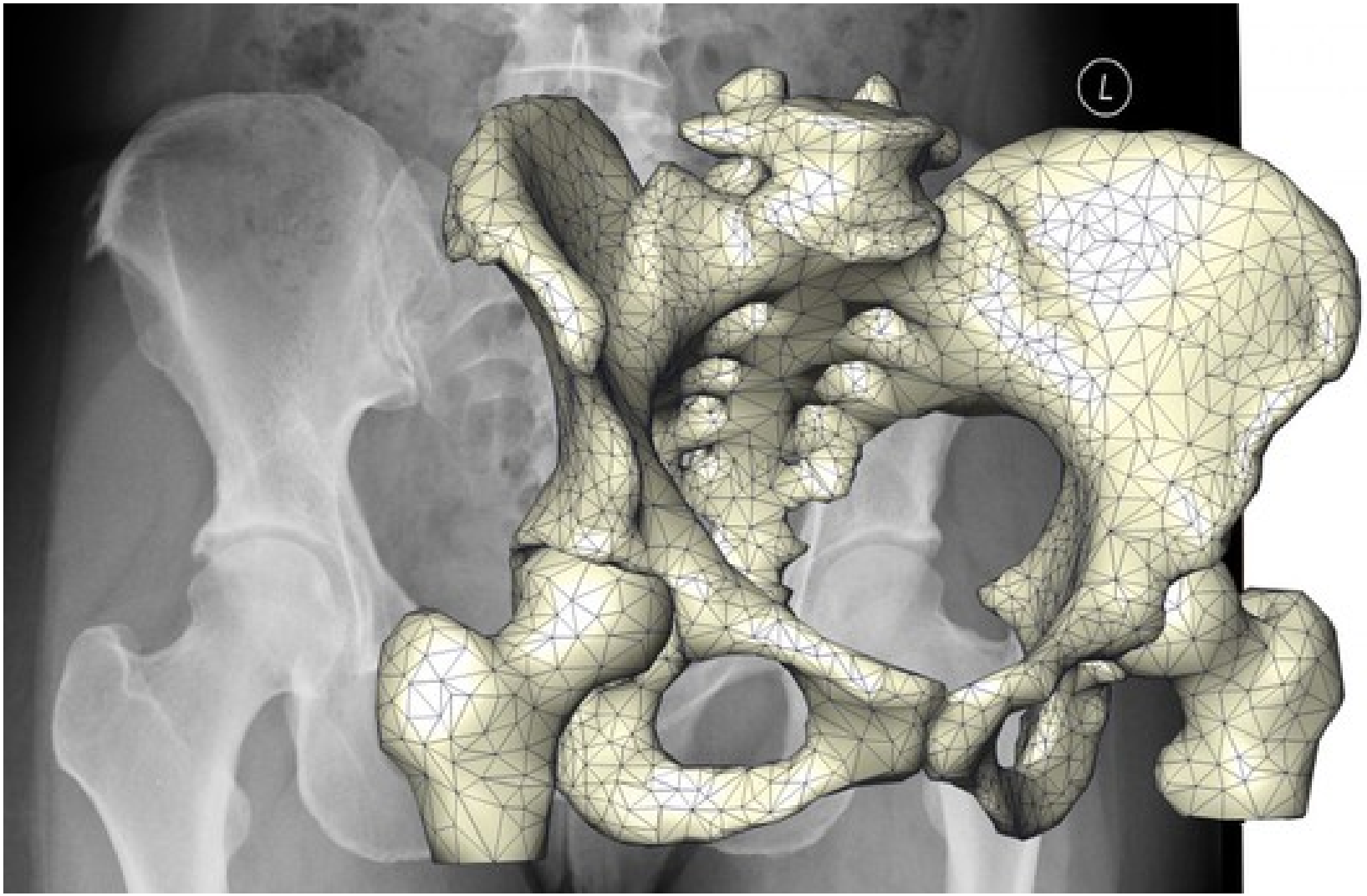
- 3D assessment of tissue and organs from non-invasive scanning
- automated medical image analysis, interpretation, and diagnosis



- 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

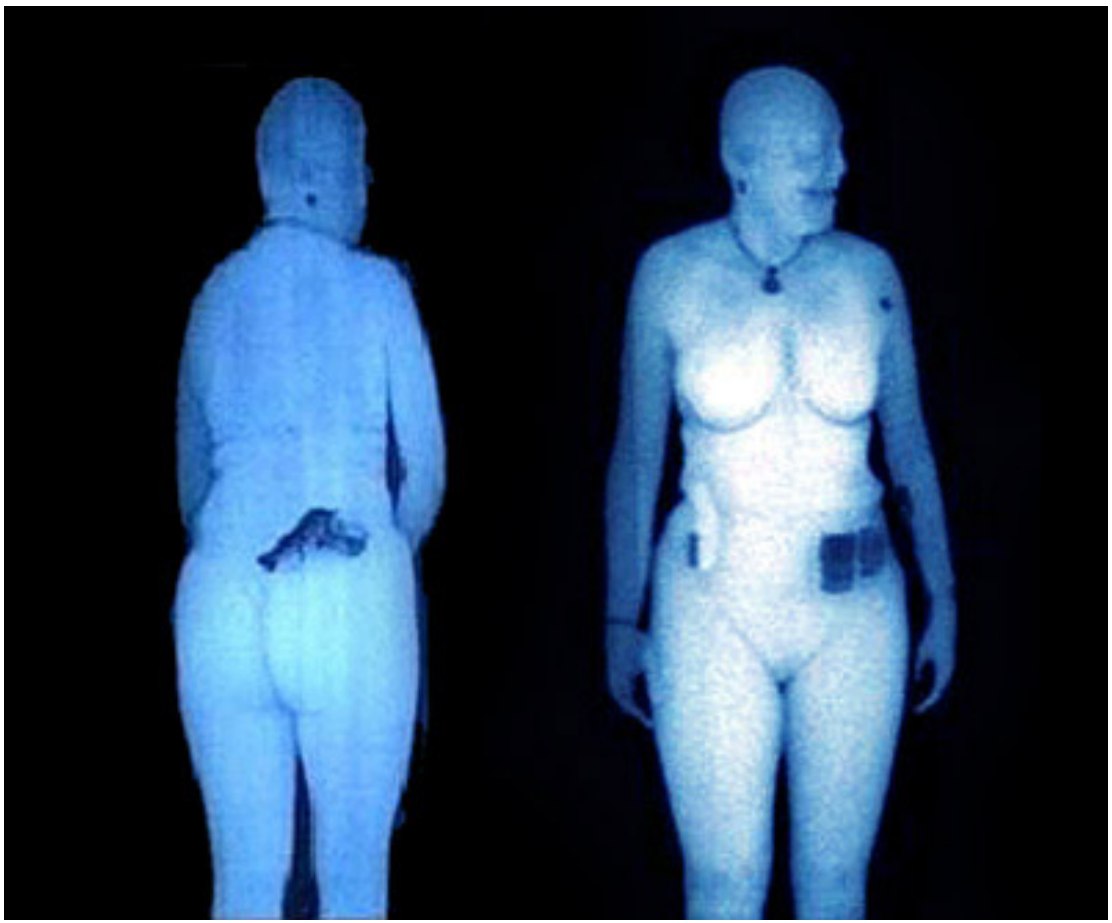


(some computer vision applications and goals, con't)

- 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



1(b). 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)*:

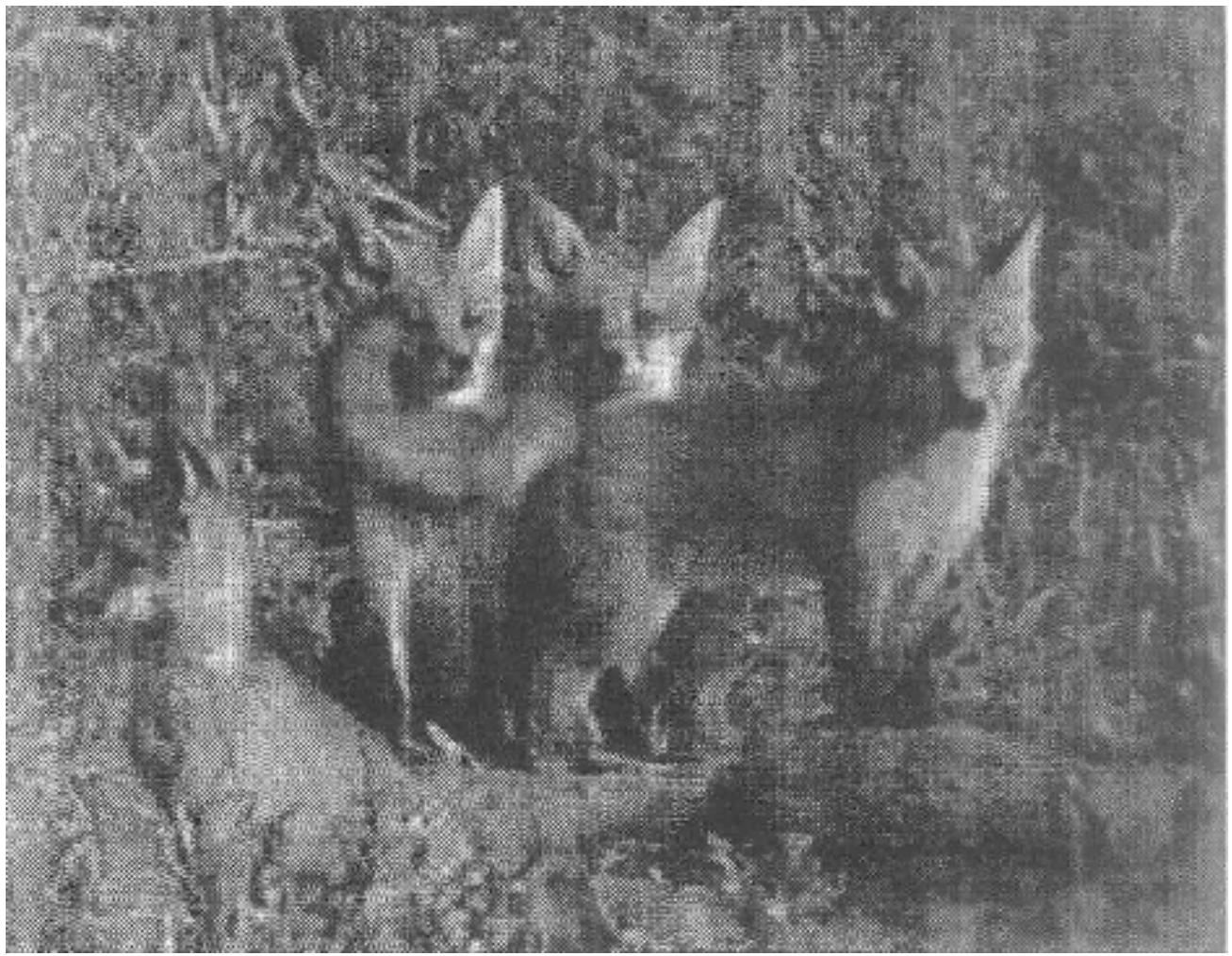
Which two pictures show the same person?

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)

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.

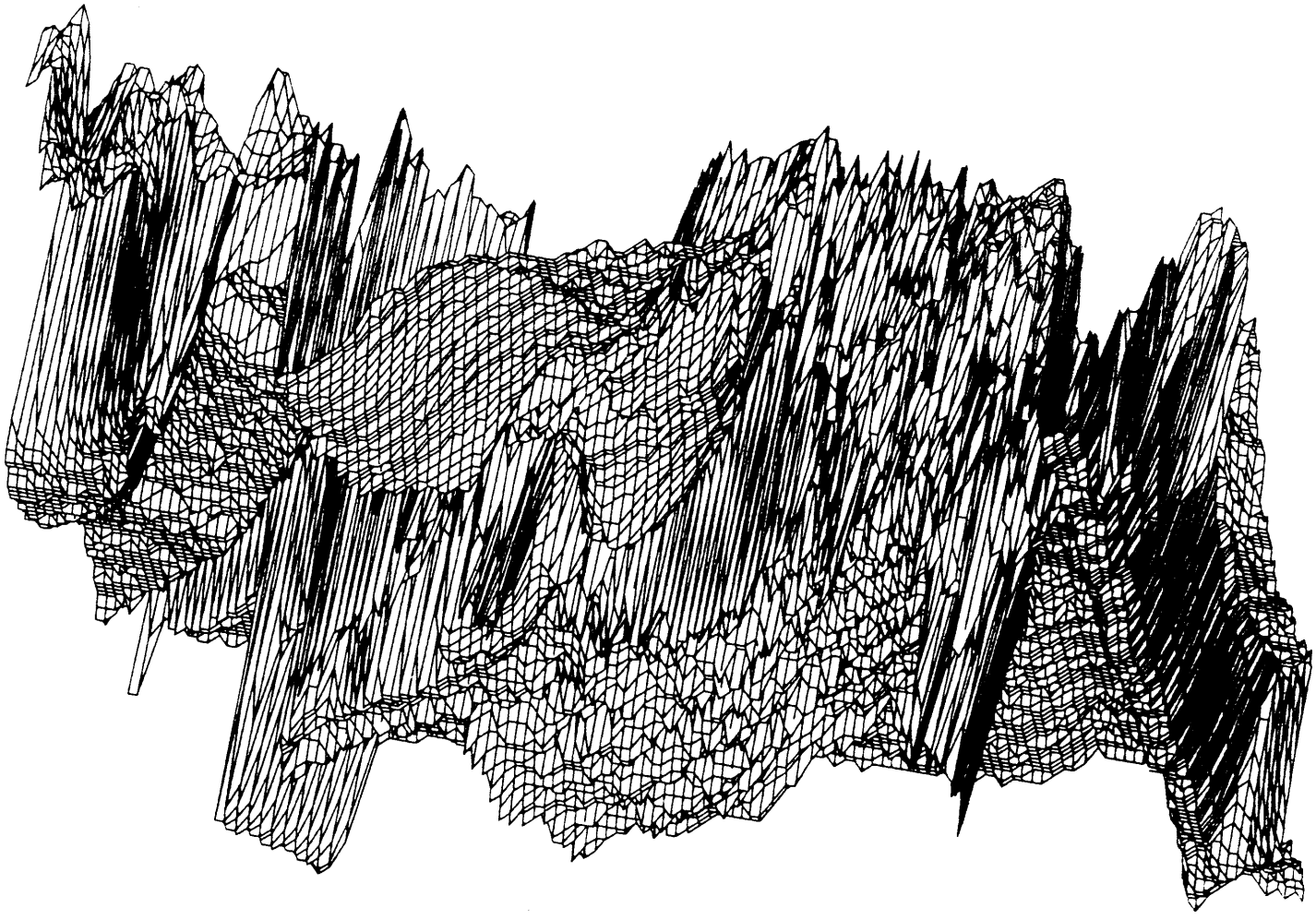
How can there possibly exist mathematical operators for such an image that can, despite its poor quality:

- 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:

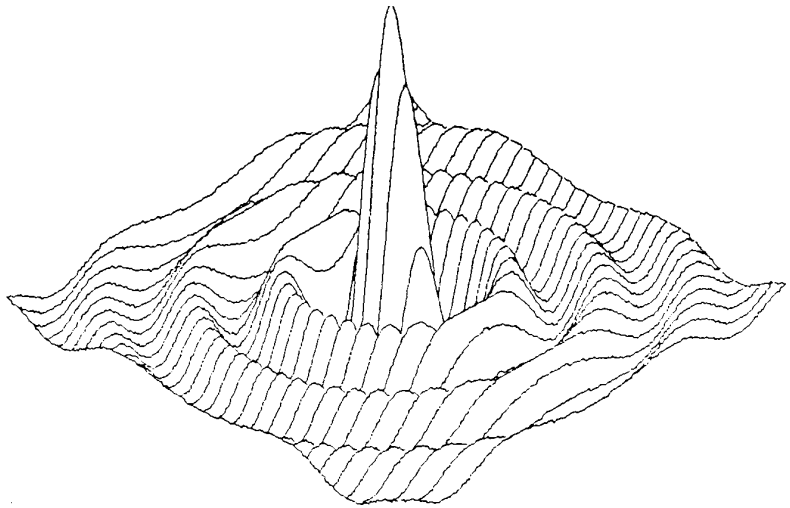
- its solution exists;
- its solution is unique;
- its solution depends continuously on the data.

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

- infering depth properties from an image
- infering surface properties from image properties
- infering colours in an illuminant-invariant manner
- infering structure from motion, shading, texture, shadows, ...

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

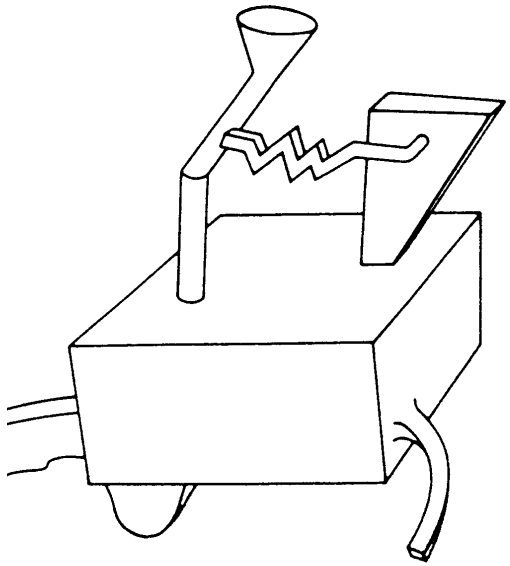
- inferring a 3D shape unambiguously from a 2D line drawing:



- 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.)

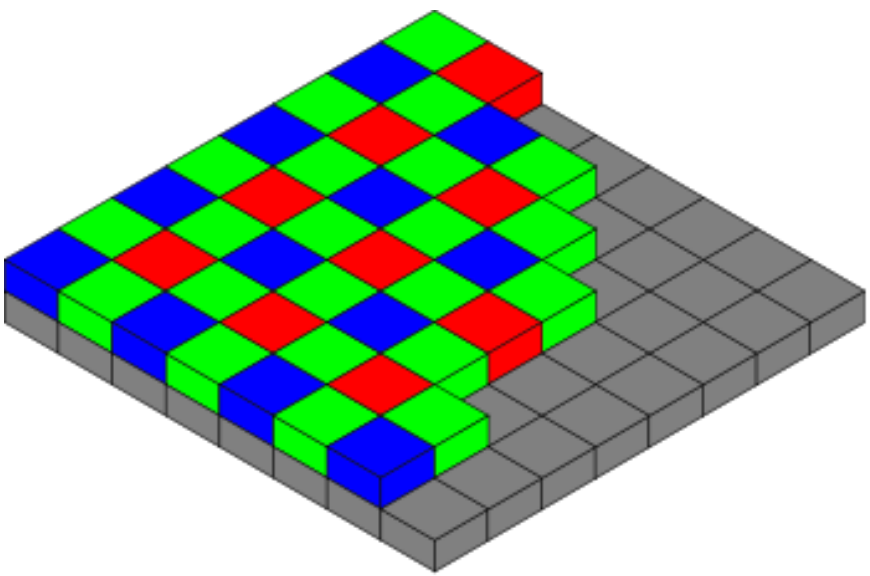
...but enough counsel of despair. Let us begin with understanding what an image array is.

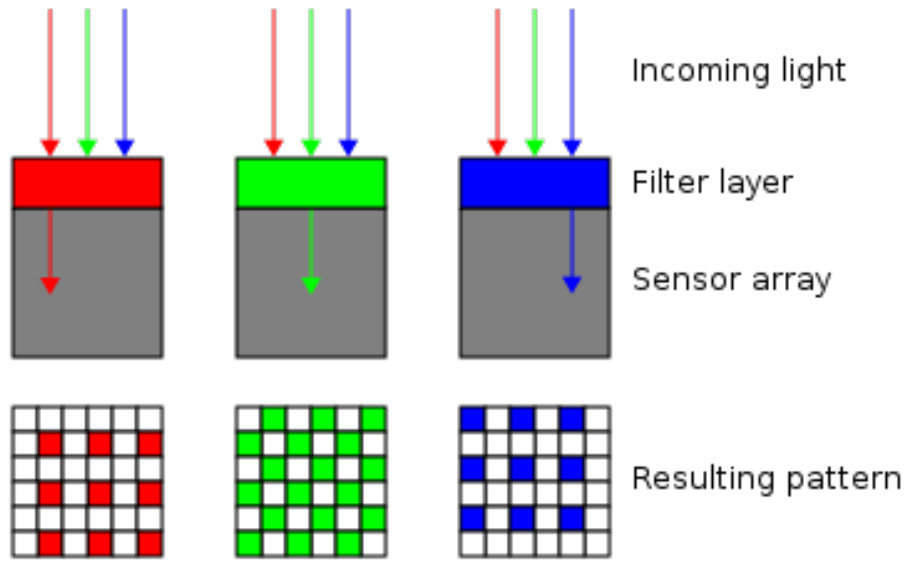
2. Image sensing, pixel arrays, CCD cameras, image coding

- A CCD video camera contains a dense array of independent sensors, which convert incident photons focused by the lens onto each point into a charge proportional to the light energy there.
- The local charge is "coupled" (hence CCD) capacitively to allow a voltage (V=Q/C) to be read out in a sequence scanning the array.
- The number of pixels (picture elements) 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.

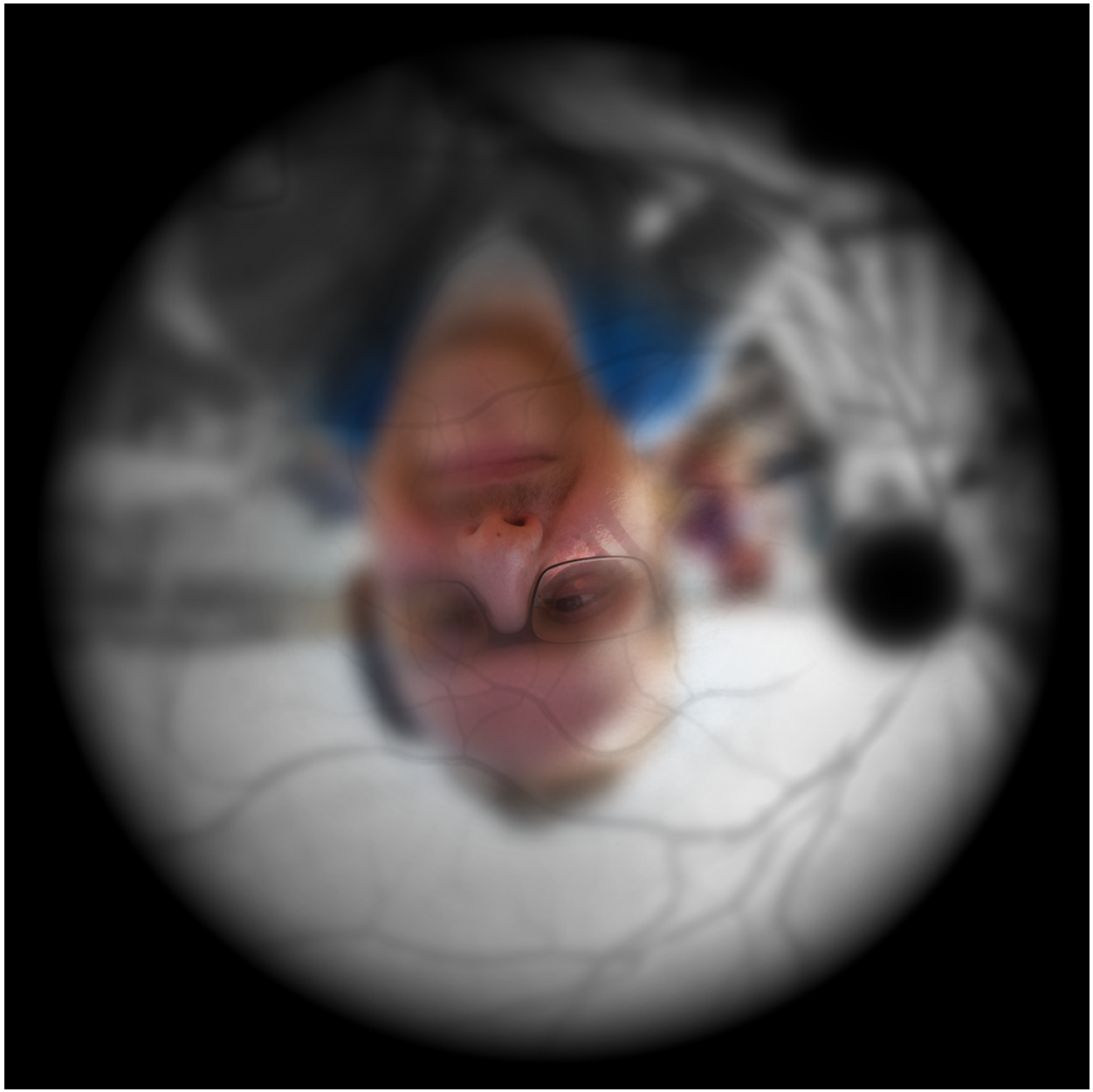
(Image formats and sampling theory, con't)

- 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* information 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.
- Because quantised image information is thus fundamentally discrete, the operations from calculus which we might want to perform on an image, like differentiation (to find edges) or integration (to perform convolutions or transforms), must be done in their discrete forms.
- The discrete form of a derivative is a finite difference. The discrete form of an integral is a (suitably normalised) summation. But it is commonplace to represent such operations using their (usually 2D) notations from continuous mathematics: $\frac{d}{dx}$, ∇^2, and $\int\int \, dx\, dy$.

3. Biological visual mechanisms: retina to visual cortex



Active Contours

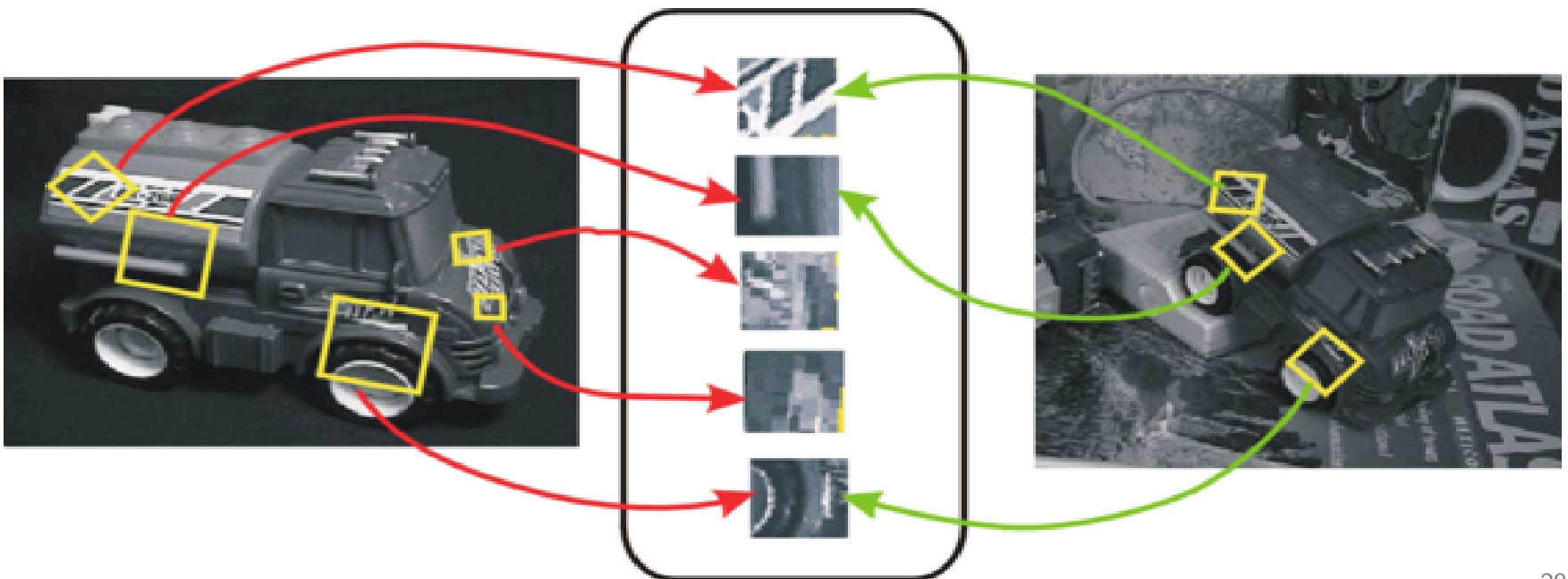
- Match a deformable model to an image, by "energy minimisation"
- Used for shape recognition, object tracking, and image segmentation
- A deformable spline (or "snake") changes its shape under competing forces: image forces that pull it towards certain object contours; and internal forces ("stiffness") 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...)

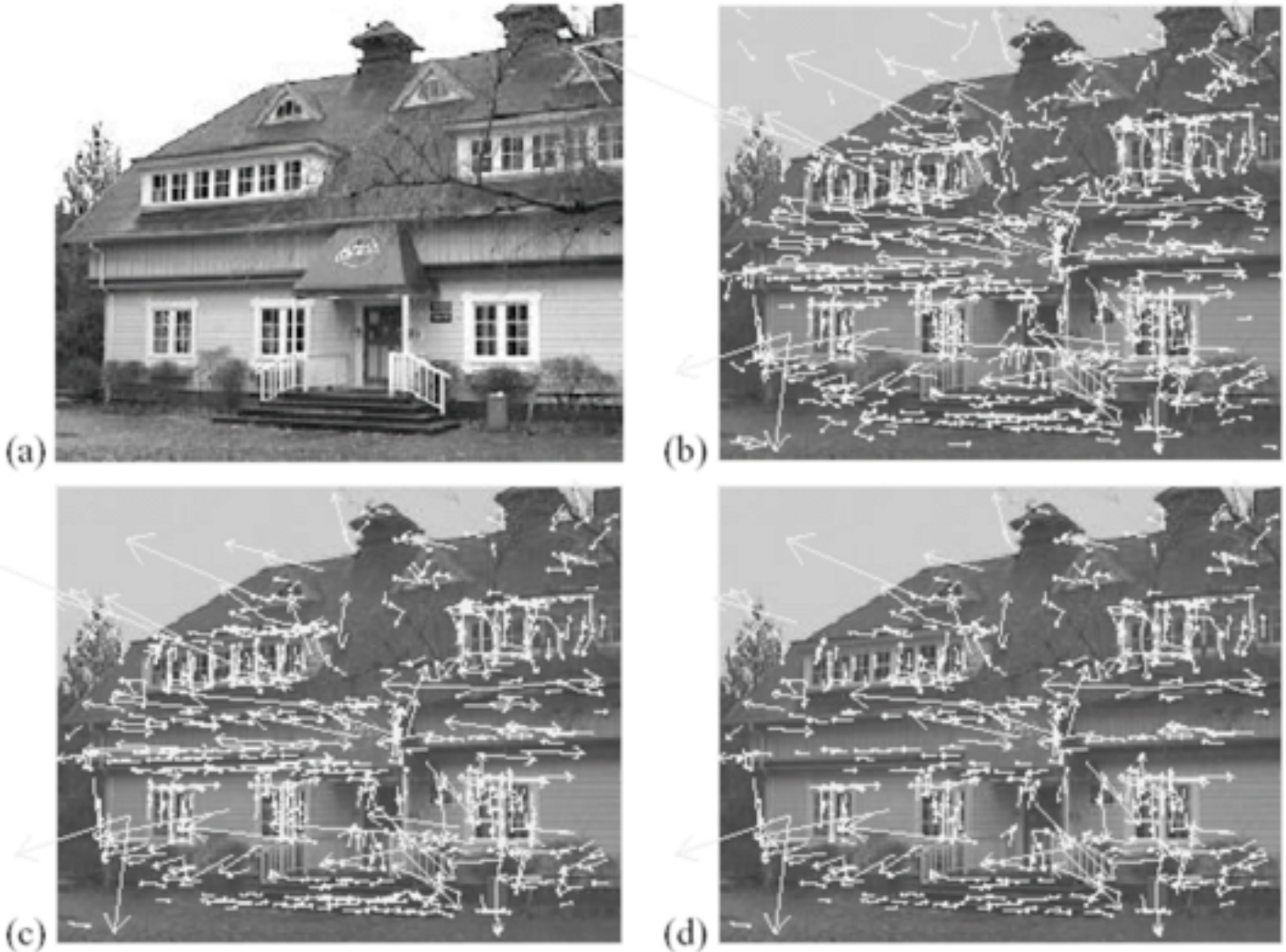


(Scale-Invariant Feature Transform, con't)

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

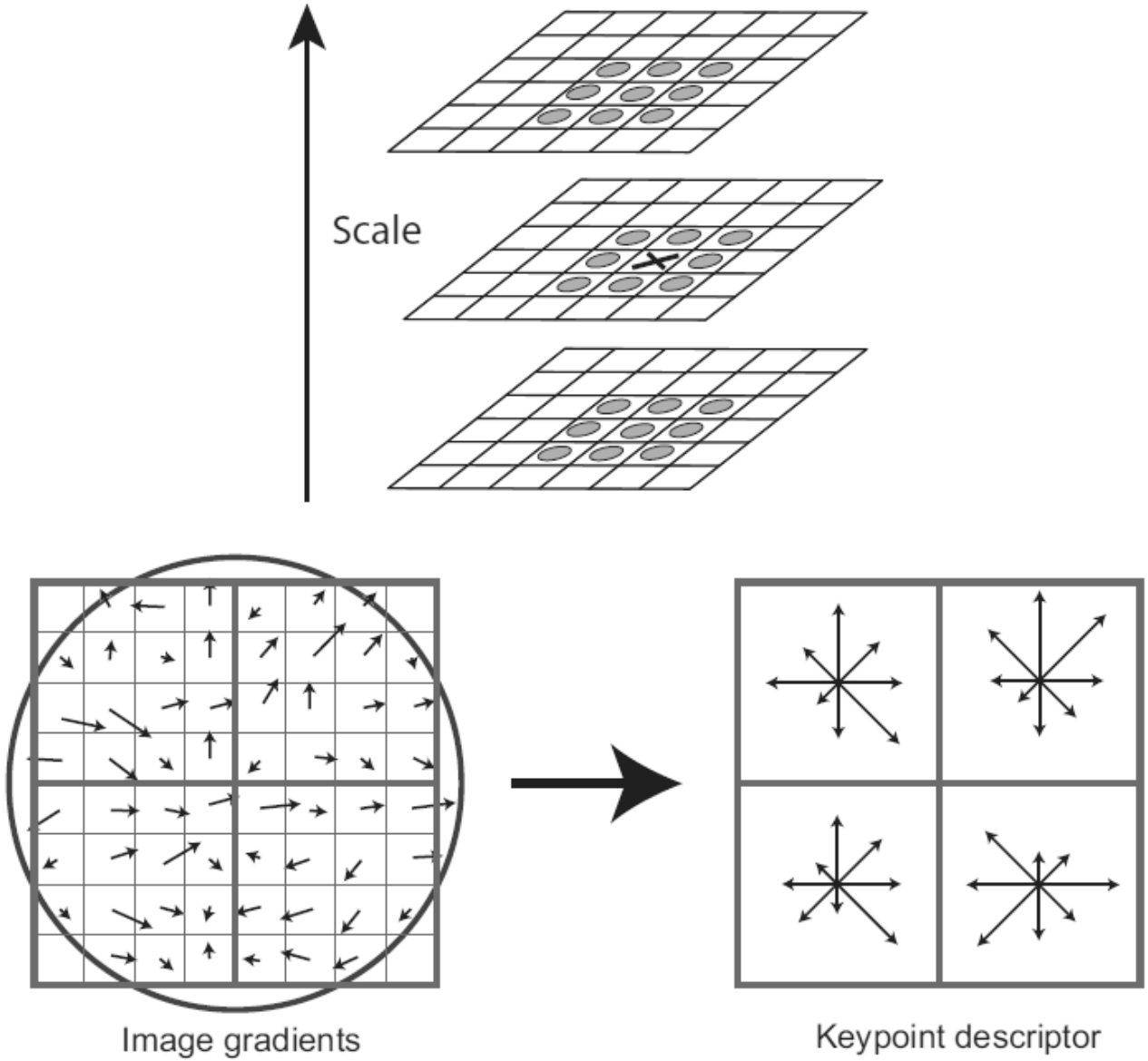
- Various kinds of feature detectors can be used, but they should have an orientation index and a scale index
- Classic approach of Lowe used extrema (maxima and minima) of difference-of-Gaussian functions in scale space
- Build a Gaussian image pyramid in scale space by successively smoothing (at octave 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

Summary: philosophy and theology of the SIFT

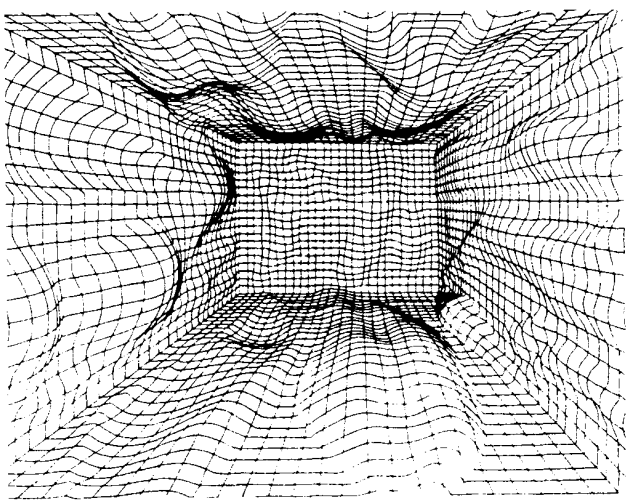
The Doctrine of Suspicious Coincidences

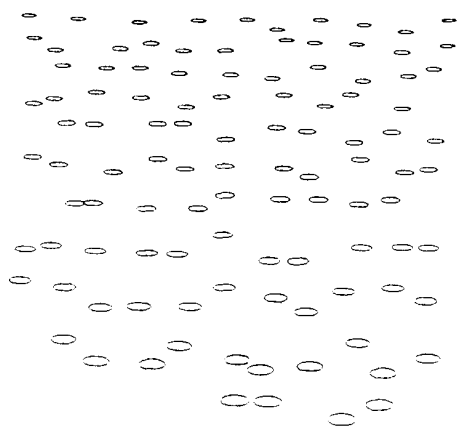
When the recurrence of patterns just by chance is a highly improbable explanation, it is unlikely to be a coincidence.

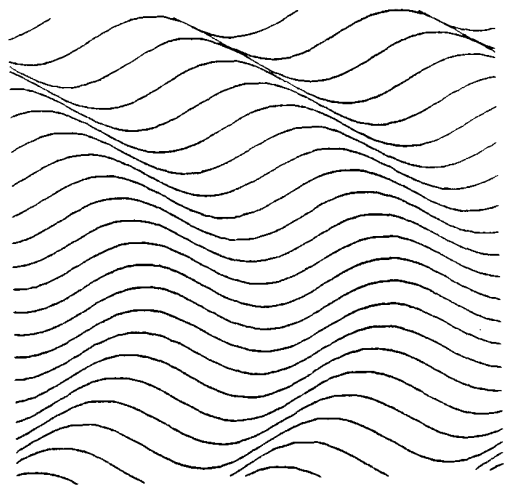
UNIVERSITY OF CAMBRIDGE

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



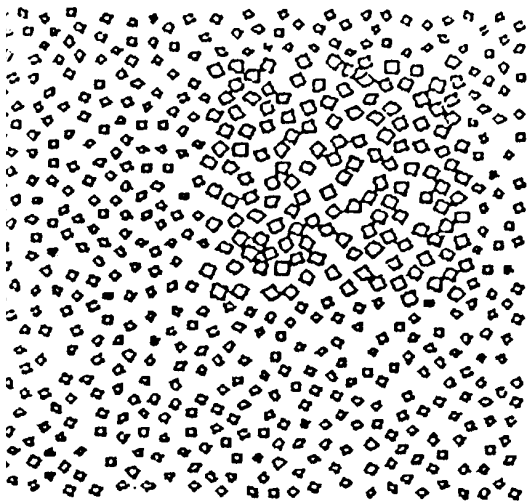
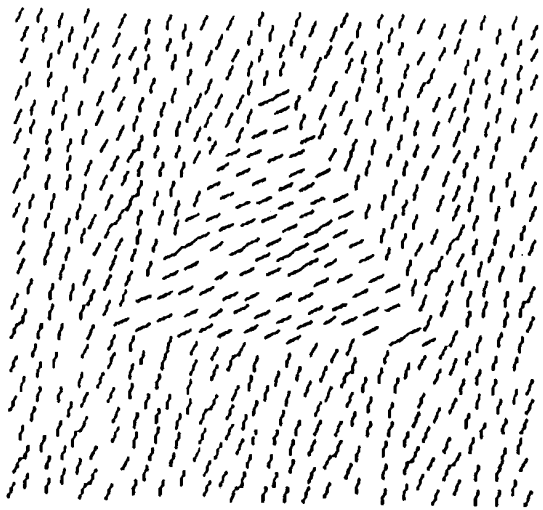
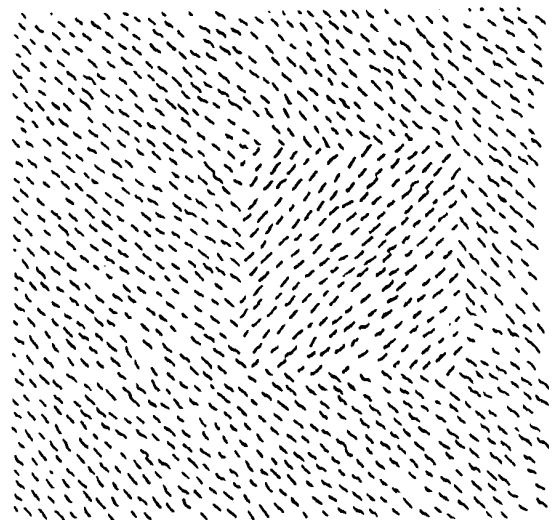




(Structure from Texture, con't)

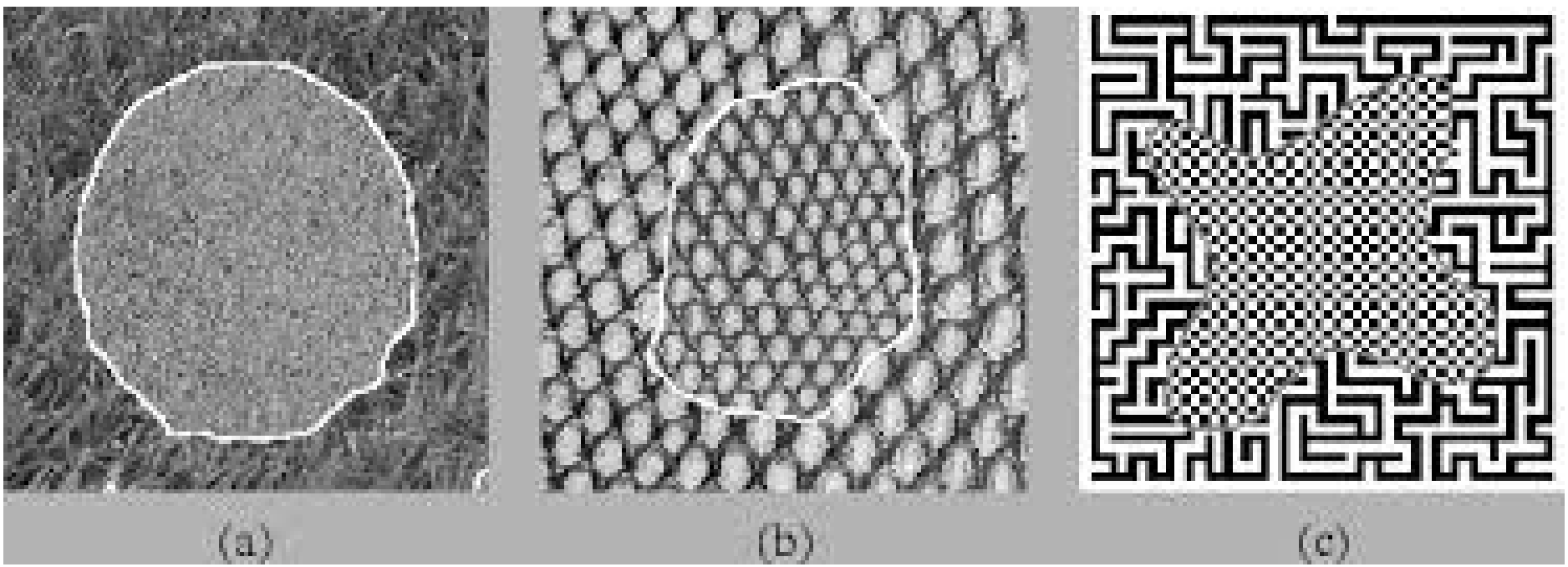
- Quasi-periodicity can be detected best by Fourier-related methods
- The eigenfunctions of Fourier analysis (complex exponentials) are periodic, with a specific scale (frequency) and wavefront orientation
- Therefore they excel at detecting a correlation distance and direction
- They can estimate the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

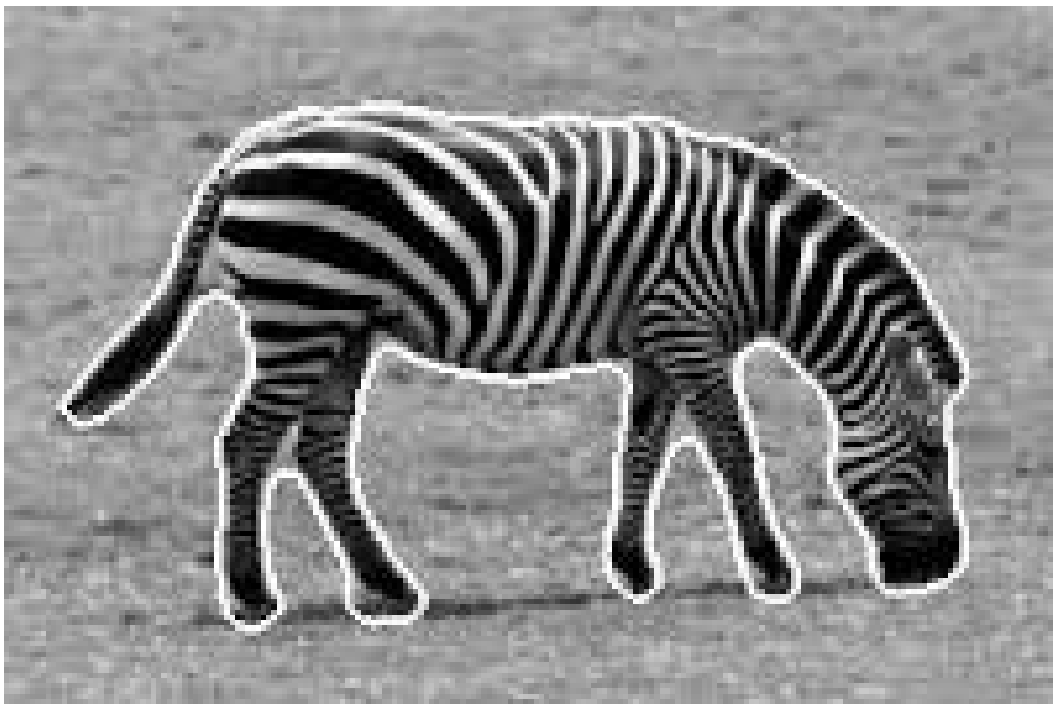
(Structure from Texture, con't)

- 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)



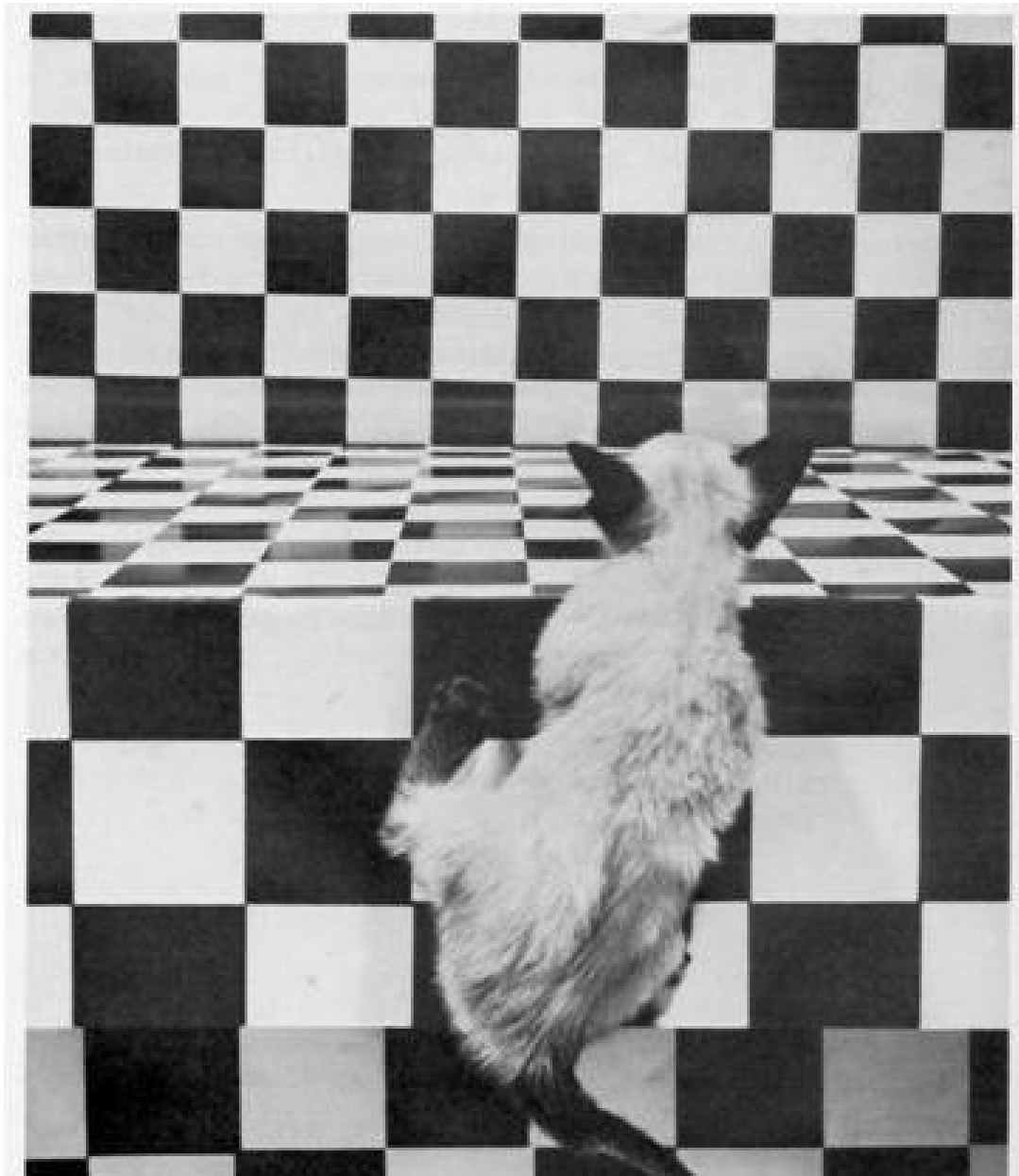
(Structure from Texture, con't)

- 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...



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

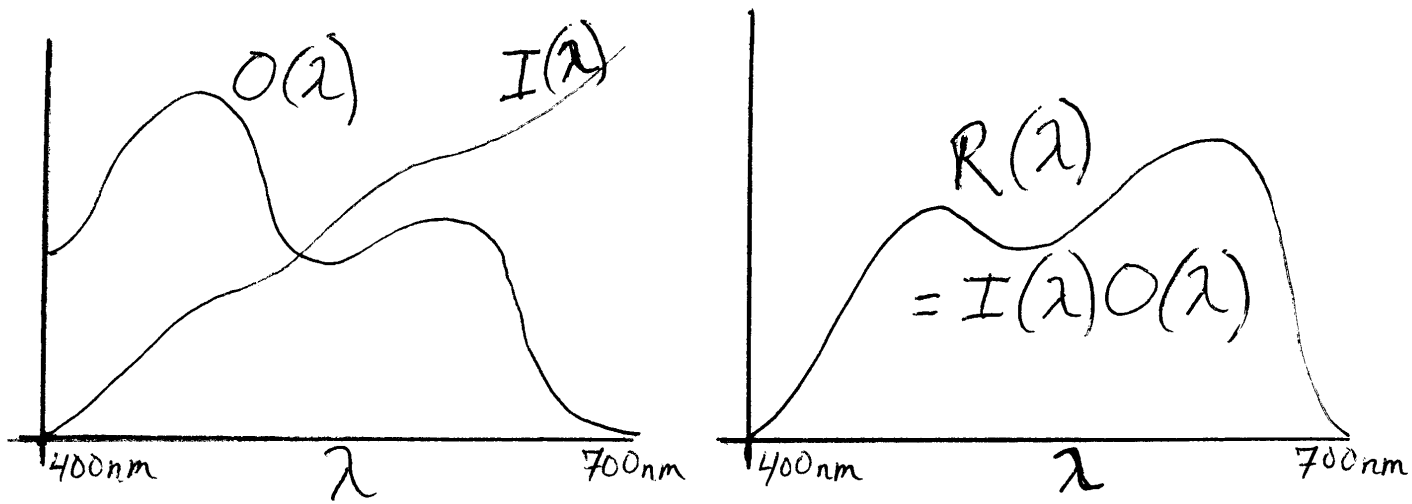
The brown tile at the centre of the illuminated upper face of the cube, and the orange tile at the centre of the shadowed front face, are actually returning the same light to the eye (as is the tan tile lying in front)

(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 $\leftarrow \lambda$)?

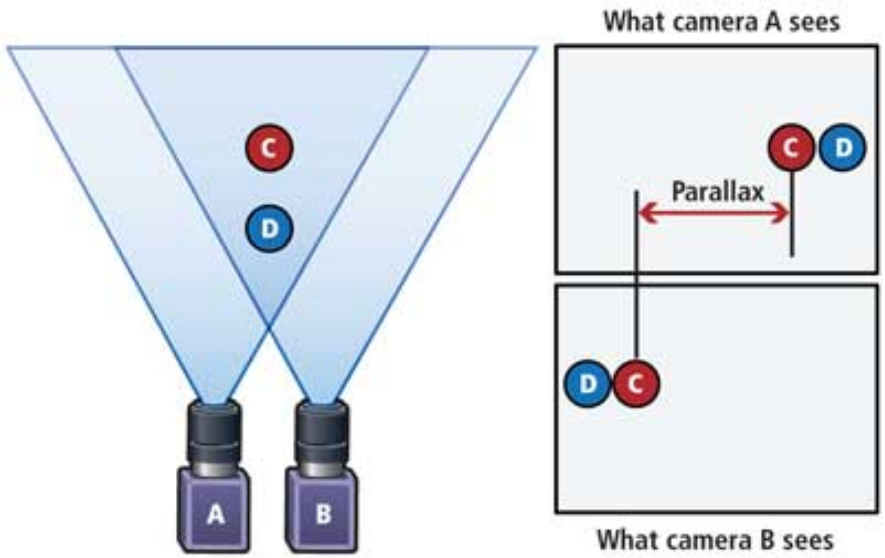
The next slide is a purely monochromatic (black-and-white) picture. But you can cause it to explode into compelling colours by re-calibrating your brain, using the *subsequent* false colour image (2 slides ahead):

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

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



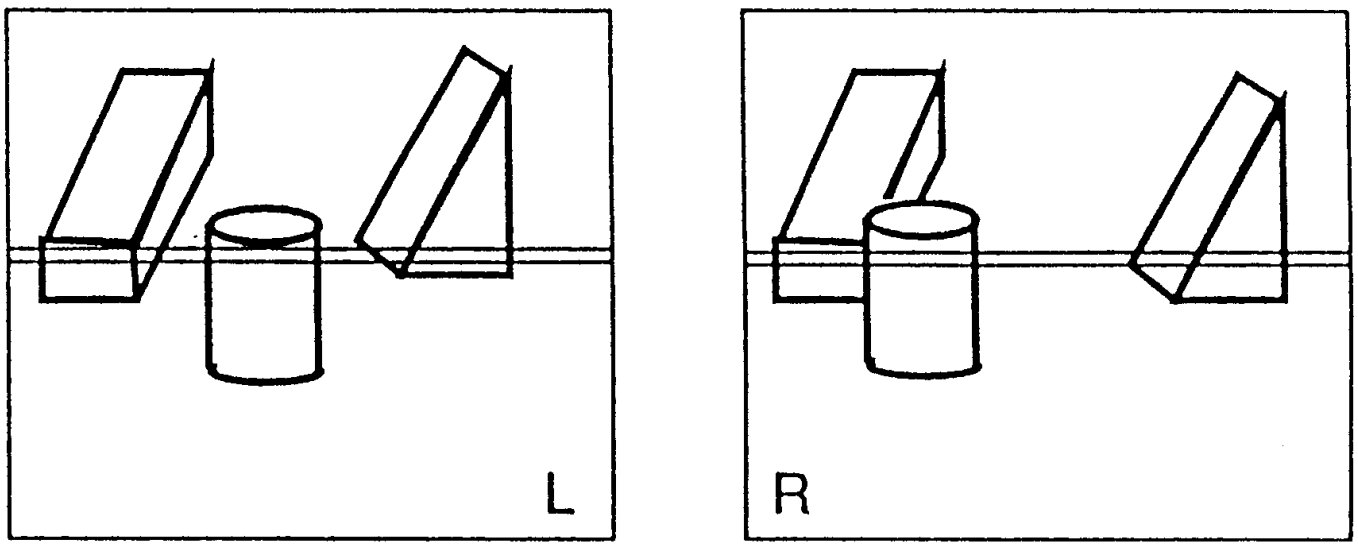
(Structure from Stereo Vision, con't)

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
- The multi-scale image pyramid 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

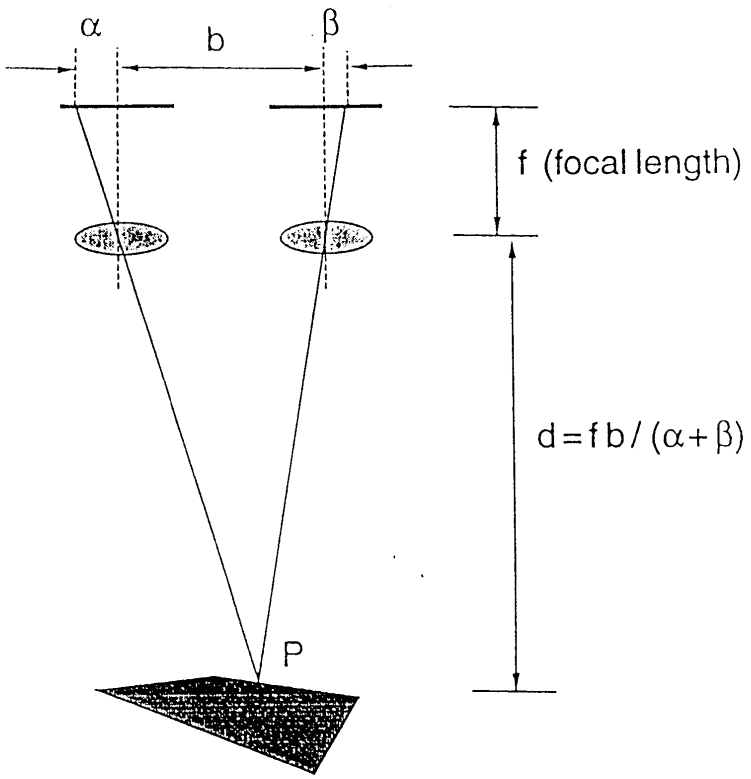


(Structure from Stereo Vision, con't)

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

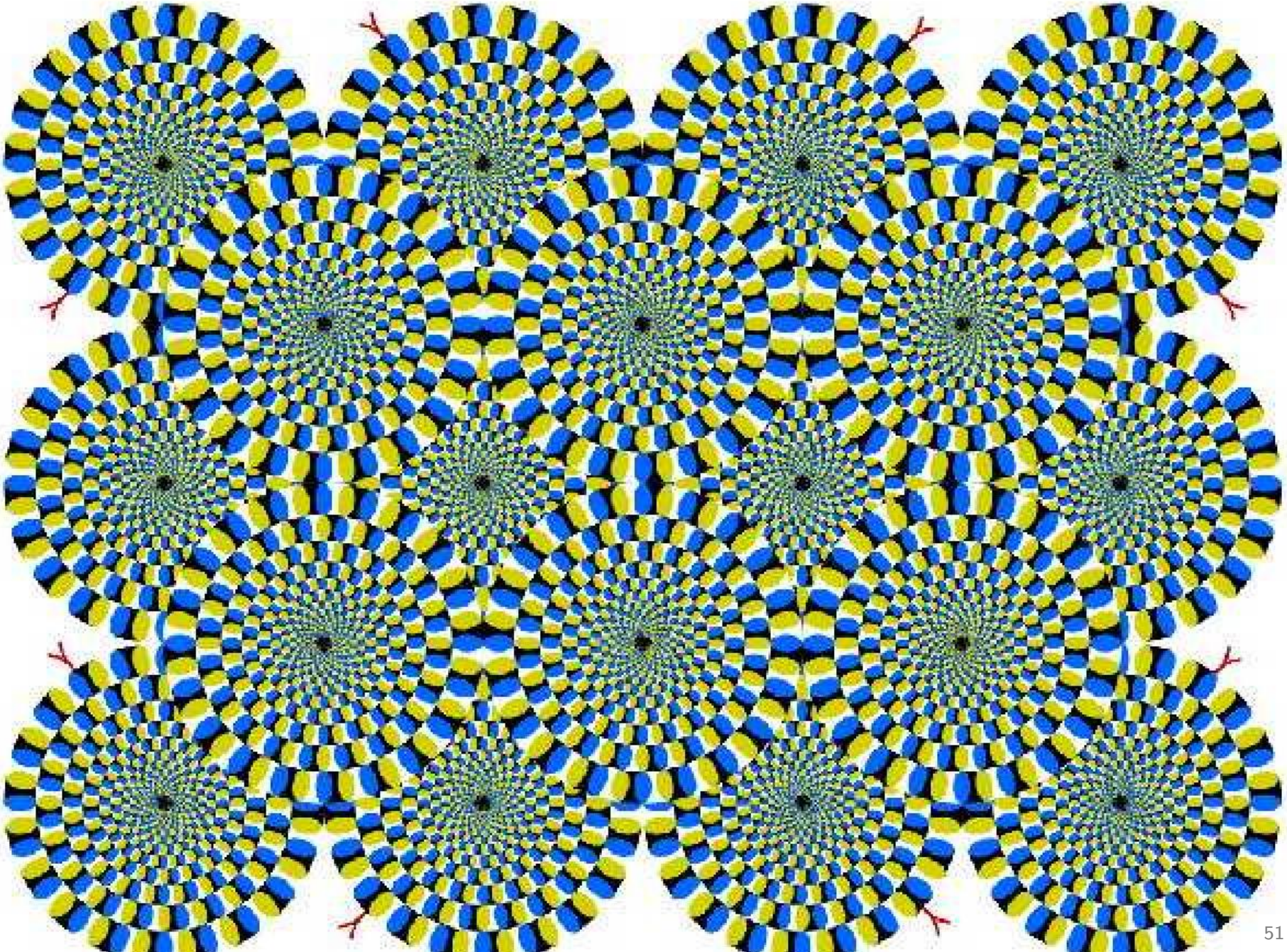


Note: P is "at infinity" if $(\alpha, \beta) = 0$

(Structure from Stereo Vision, con't)

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

Functional streaming: colour and motion pathways



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.

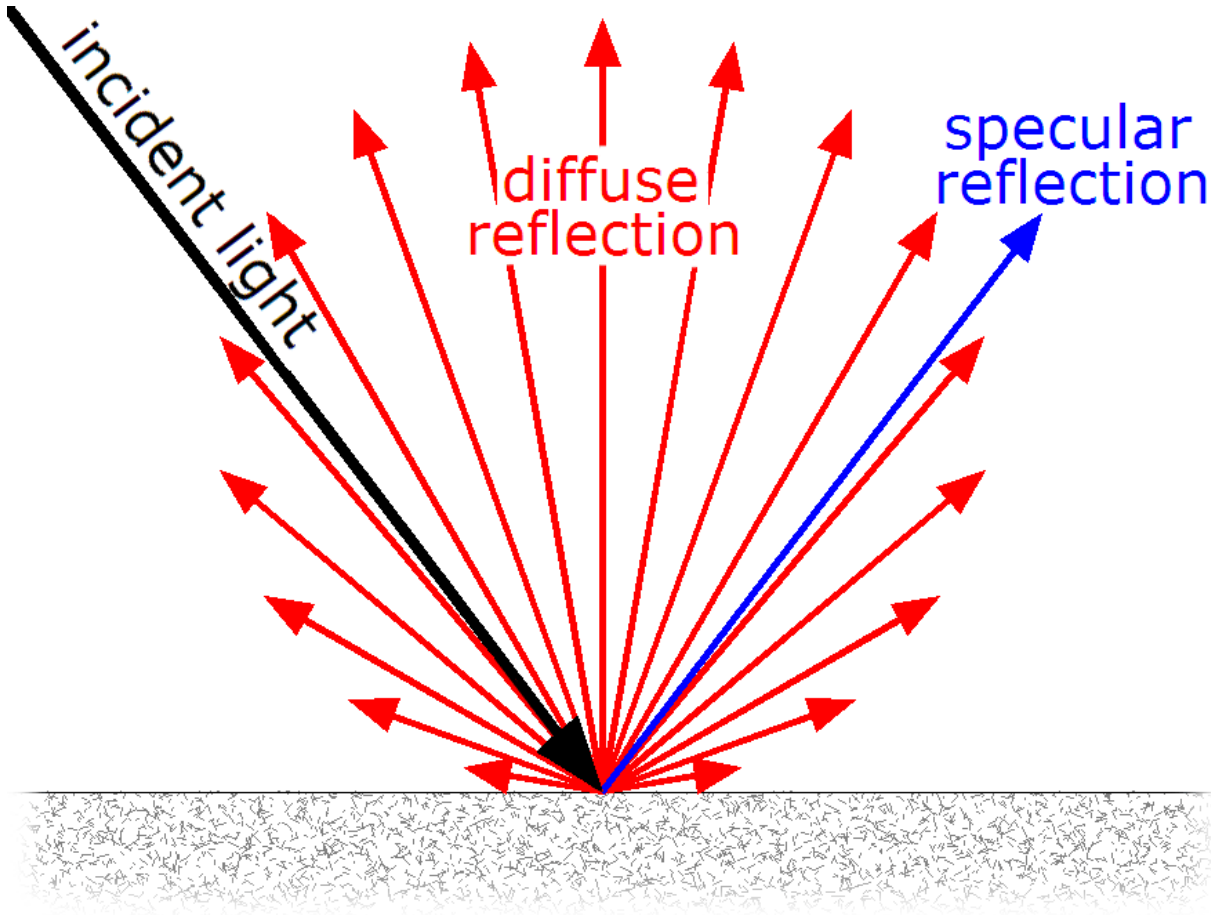
- Surface albedo is the fraction of the illuminant that is re-emitted from a surface in all directions. Thus it relates to how "light" or "dark" is the surface, and this may vary locally across it
- The amount of light reflected is the product of two factors: the surface albedo, times a geometric factor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

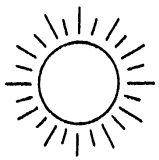


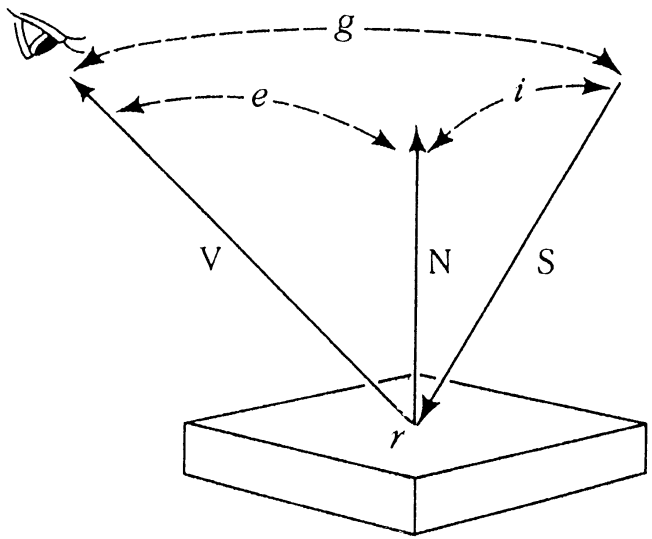
(Surfaces and Reflectance Maps, con't)

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)

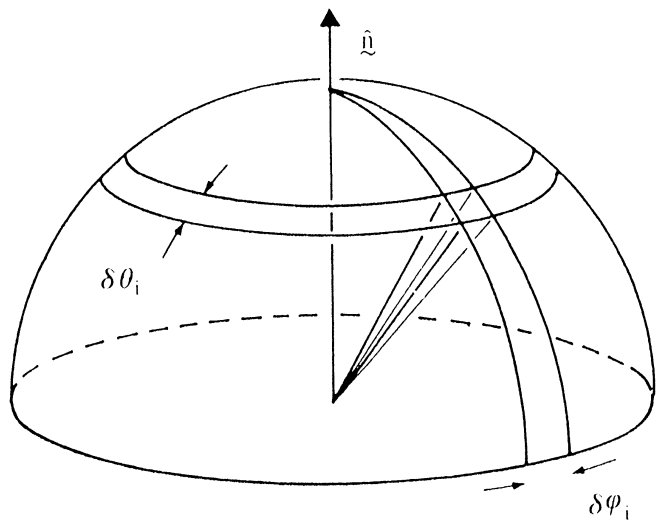
Typically, surfaces have both specular and matte properties. For example, facial skin may vary from Lambertian (powdered) to specular (oily). The purpose of powdering one's face is to specify s and n in this expression:

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

- 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



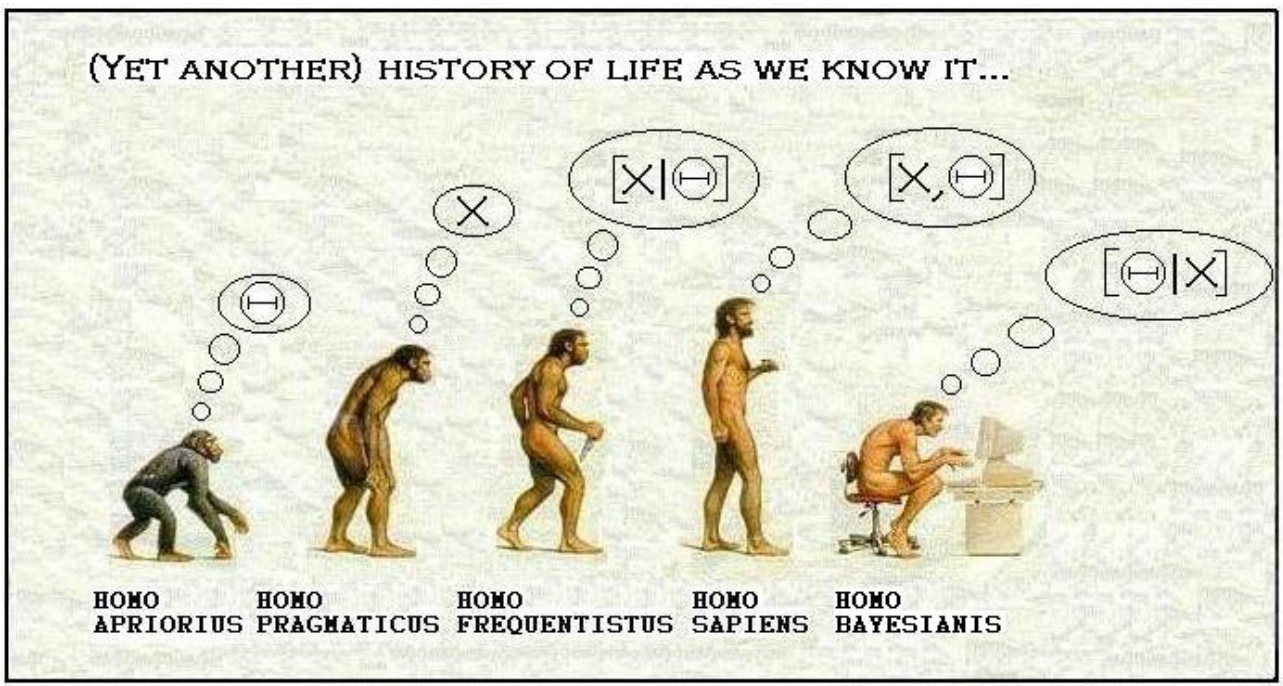
Clearly, the only way to infer the nature and geometry of surface properties from image properties is to build-in certain assumptions about the nature of the surfaces from other kinds of evidence. This requires us to consider the general problem of inference and integration of evidence

(Surfaces and Reflectance Maps, con't)

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

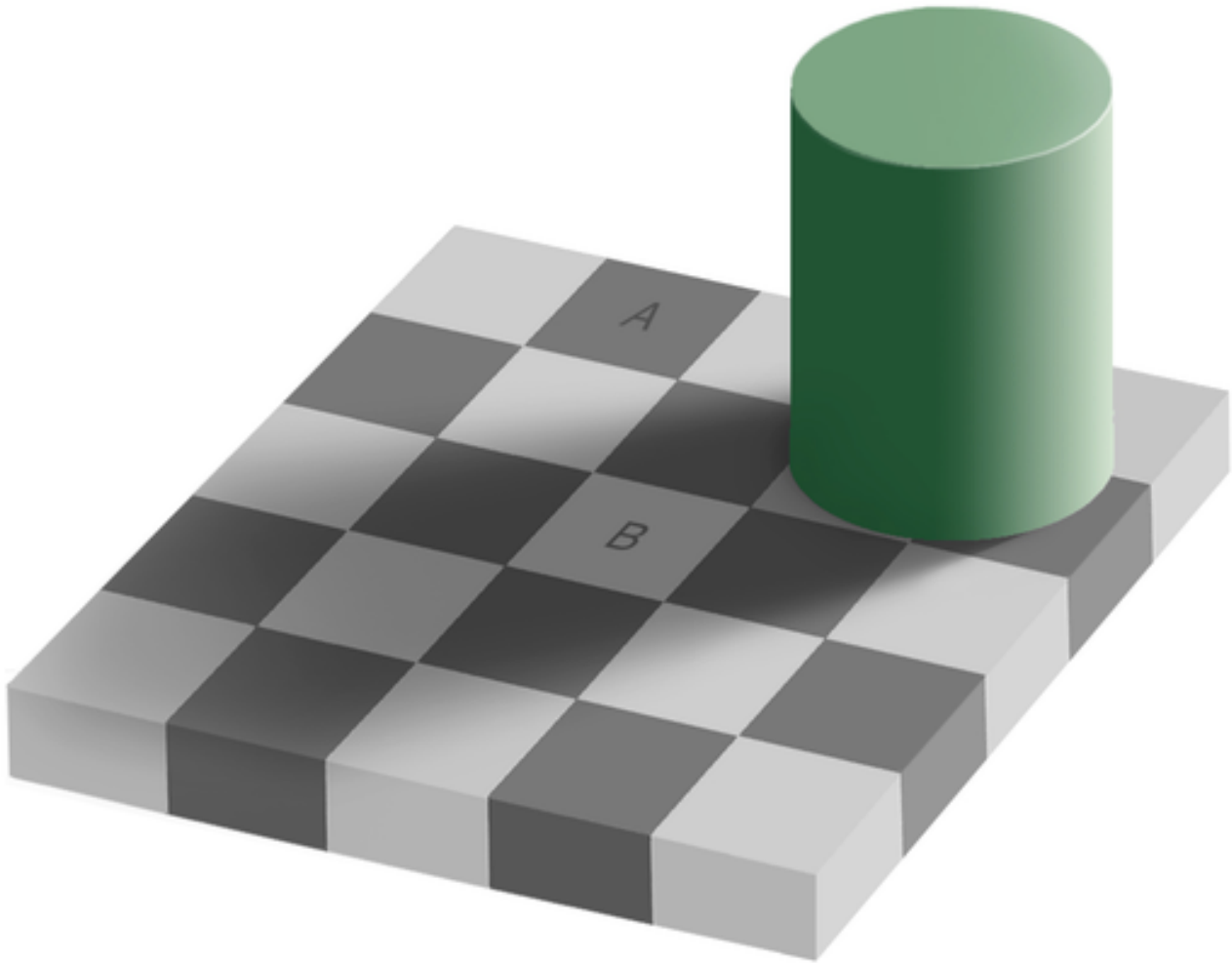
- geometry of the illuminant (e.g. is it a point source, or extended? If a point source, where is it?) Are there several light sources?
- reflectance properties of the surface. What is its reflectance map?
- geometry of the surface (its underlying shape). Are shadows cast?
- rotations of the surface 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:



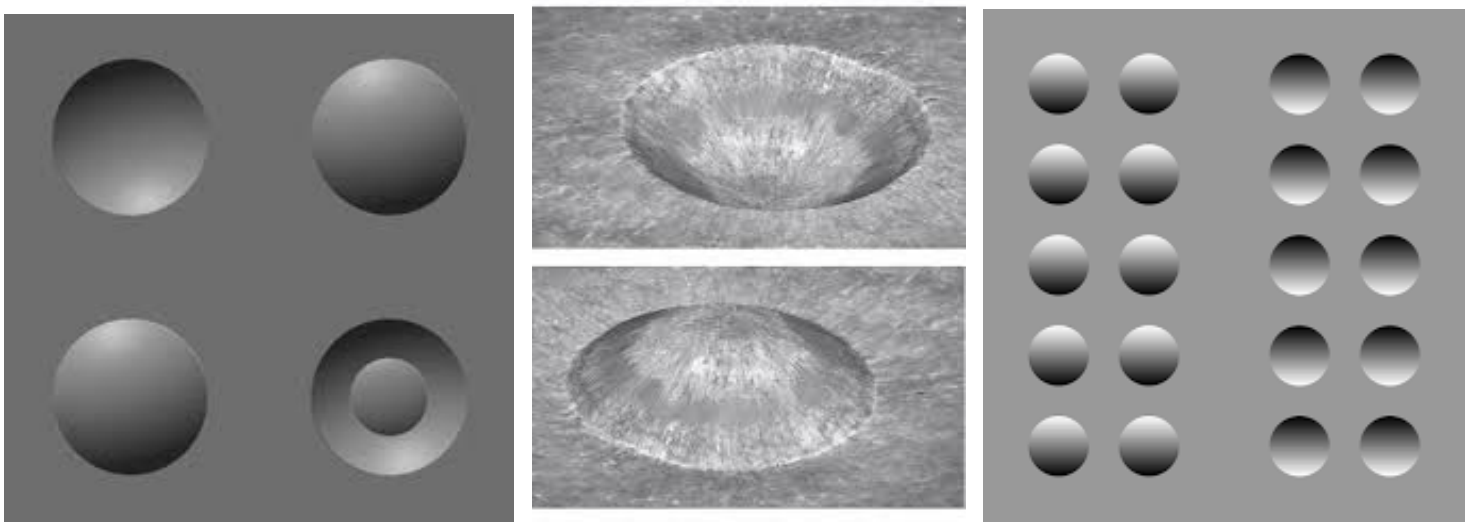
(Surfaces and Reflectance Maps, con't)

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.



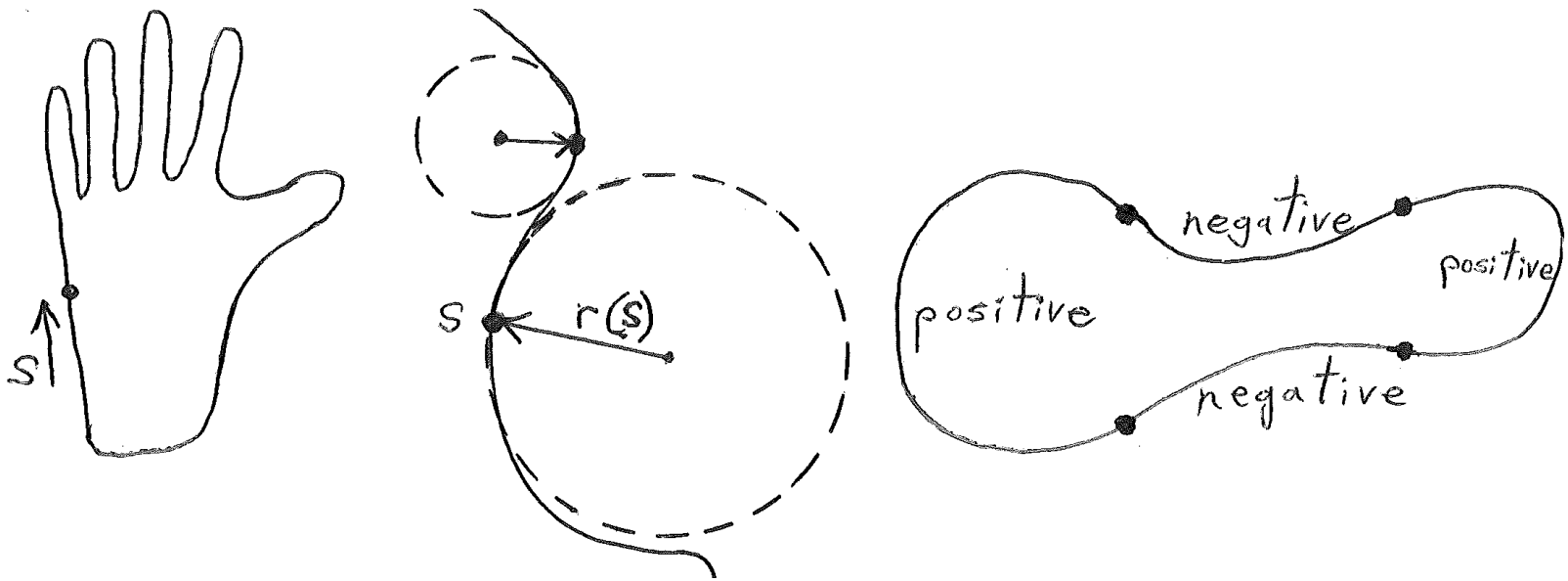
It is almost as though the assumptions (like angle of illumination) are more important than the available image data. The computational nature of the inference task then becomes one of *constraint satisfaction*. Often there are rivalrous (e.g. is it a dome or a crater?) alternative solutions: `http://www.michaelbach.de/ot/fcs_hollow-face/index.html`

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):

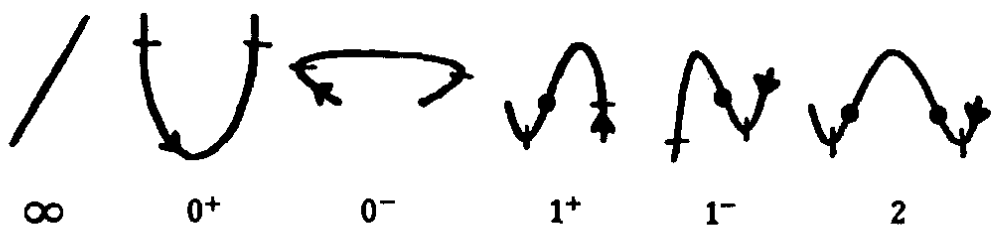
$$\theta(s) \rightarrow K\theta(s)$$

(Shape Representation and Codon Shape Grammars, con't)

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).

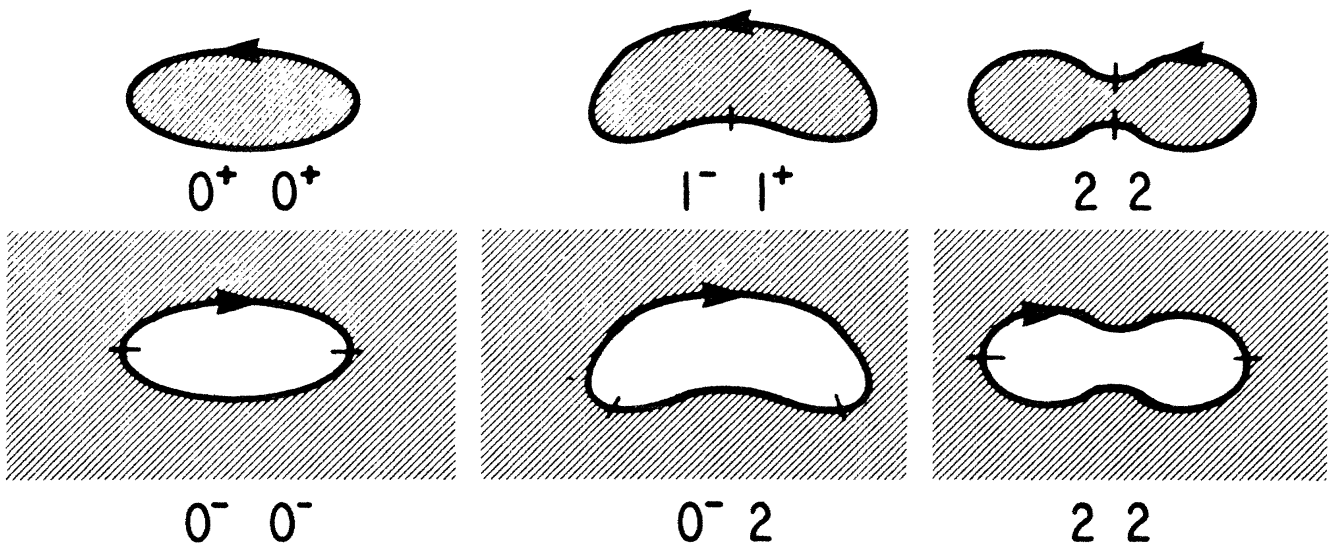
There are just six primitive codon types: all curve segments lying between minima of curvature must have 0, 1 or 2 points of zero curvature, further classified by whether a zero is encountered before ("$-$") or after ("$+$") reaching the maximum curvature in the chosen direction of traversal. Dots show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



(Shape Representation and Codon Shape Grammars, con't)

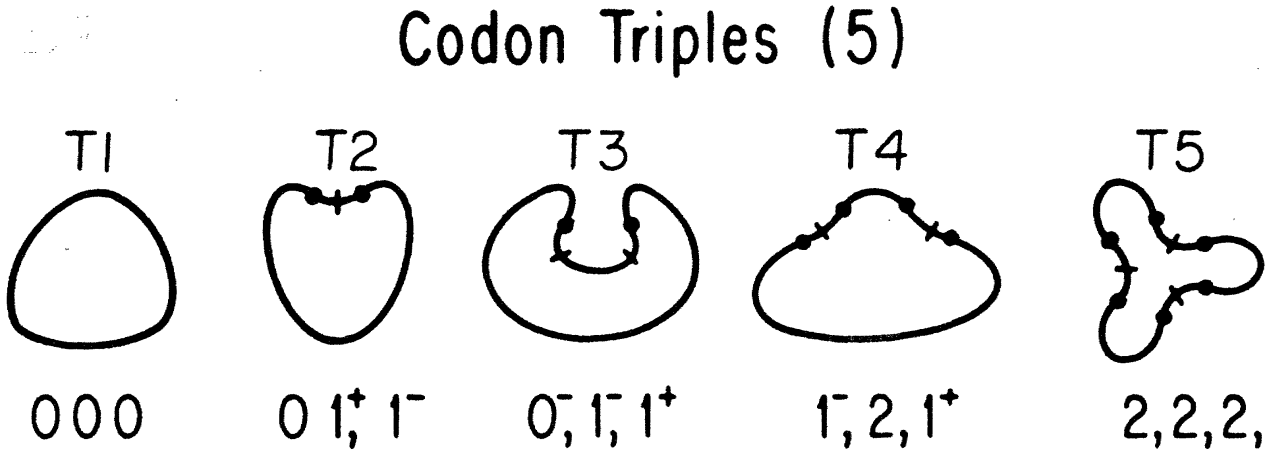
Note that because curvature is a signed quantity, the loci of its minima depend on what we take to be "figure" vs "ground". For open contours like these face profiles (alternatively Rubin's Vase profiles), if we regard "figure" as "to left", then loci of minima depend on direction of traversal:

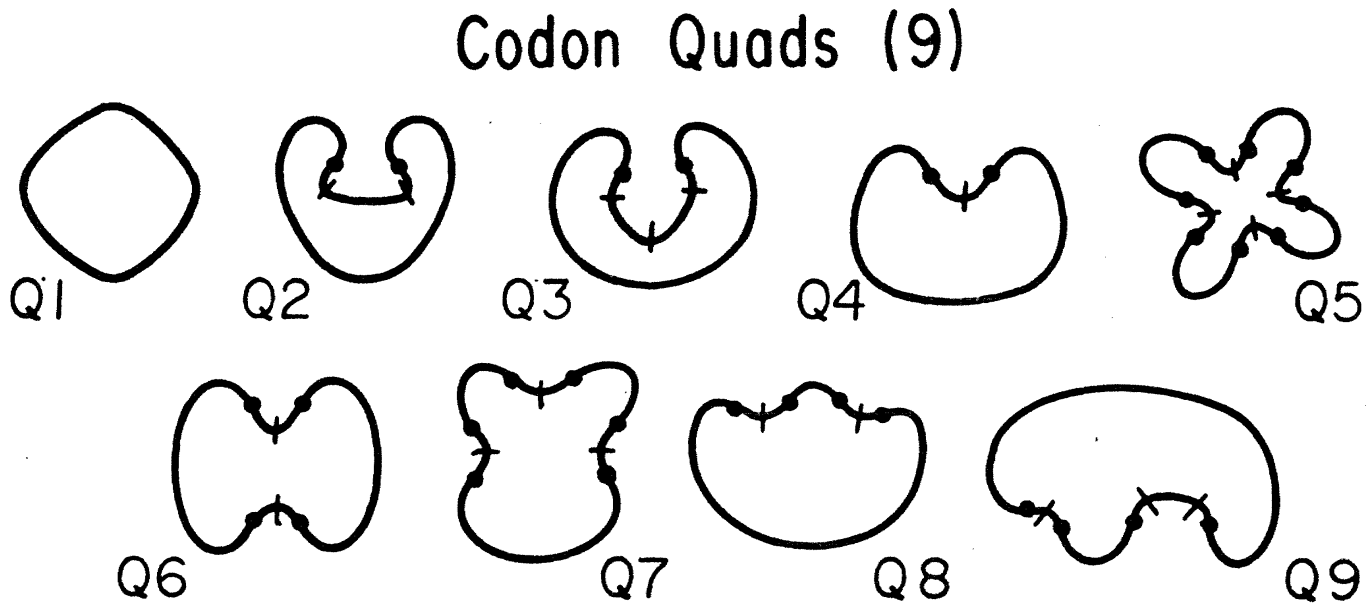
There are 3 possible Codon Pairs (string type depending on direction):



(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.

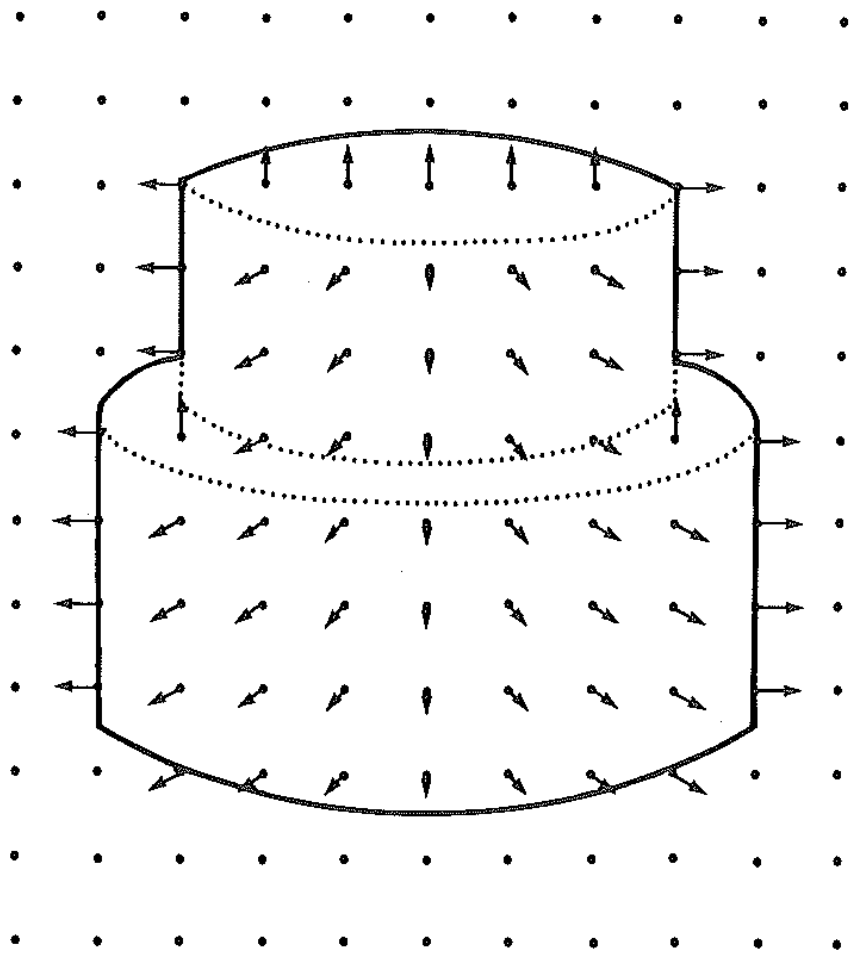
The inflexion (a zero of curvature) of a 3D curve is preserved under 2D projection, thereby guaranteeing that the ordinal relations among the extrema of curvature will also be preserved when projected to an image.

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

One scheme for bridging the gap between 2D image (appearance-based) and 3D model-based descriptions is called the "2.5-dimensional sketch". Surface normals are computed and assigned to each point in the image, like a pin-cushion, indicating 3D shape.



(Volumetric Descriptions of 3D Shape, con't)

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:

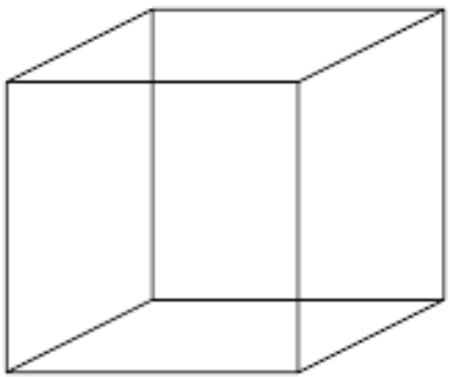
- cylinders: $(\alpha, \beta, \gamma) = (2, 2, 100)$ and $A = B$
- rectangular solids: $(\alpha, \beta, \gamma) = (100, 100, 100)$
- prolate spheroids (shaped like zeppelins): $(\alpha, \beta, \gamma) = (2, 2, 2)$ and (say) $A = B$ but $C < (A, B)$
- oblate spheroids (shaped like tomatoes): $(\alpha,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

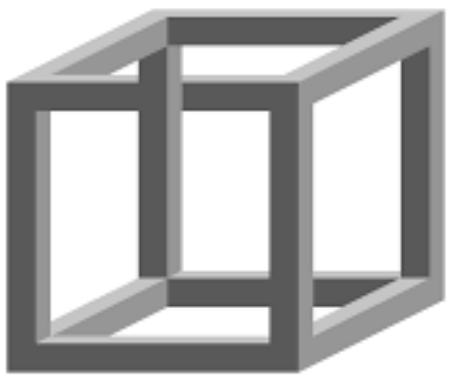
Rotations of such objects in 3D produce cross-terms in (xy, xz, yz). Parameters (A, B, C) determine object dimensions. Origin-centred.

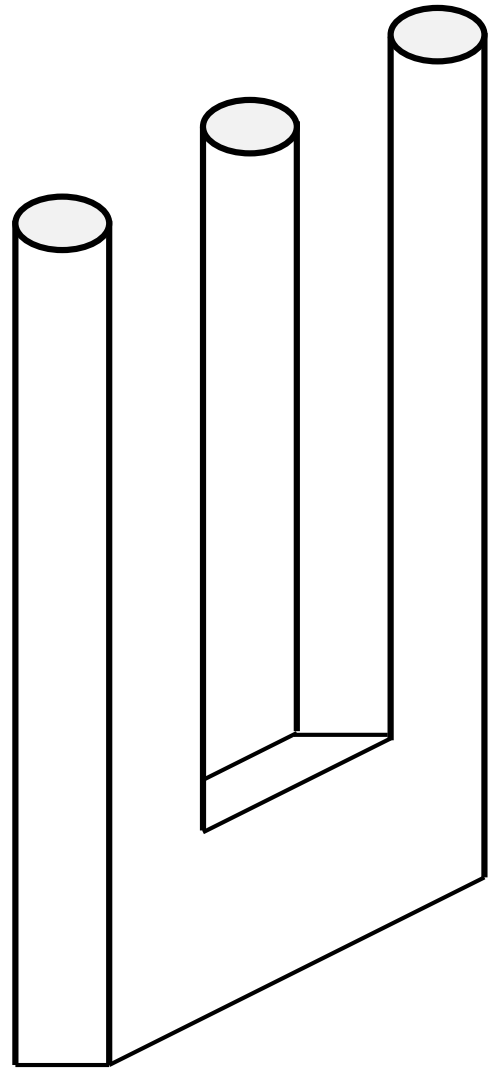
These simple, parametric models for solids, augmented by Boolean relations for conjoining them, allow the generation of object-centered, "volumetric" descriptions of many objects (instead of an image-based description) by just listing parameters $(\alpha, \beta, \gamma, A, B, C)$ and relations, rather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

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?

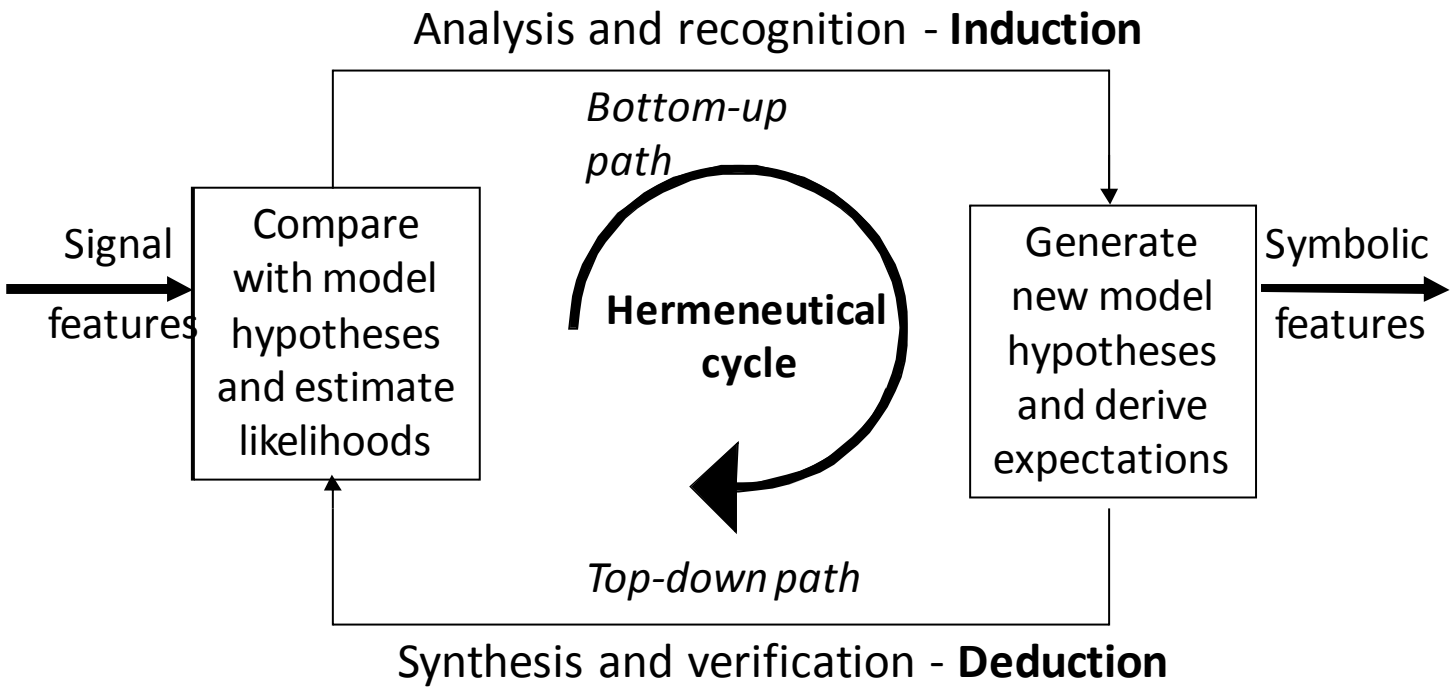






Vision as Perceptual Inference and Hypothesis Testing

- 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



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.

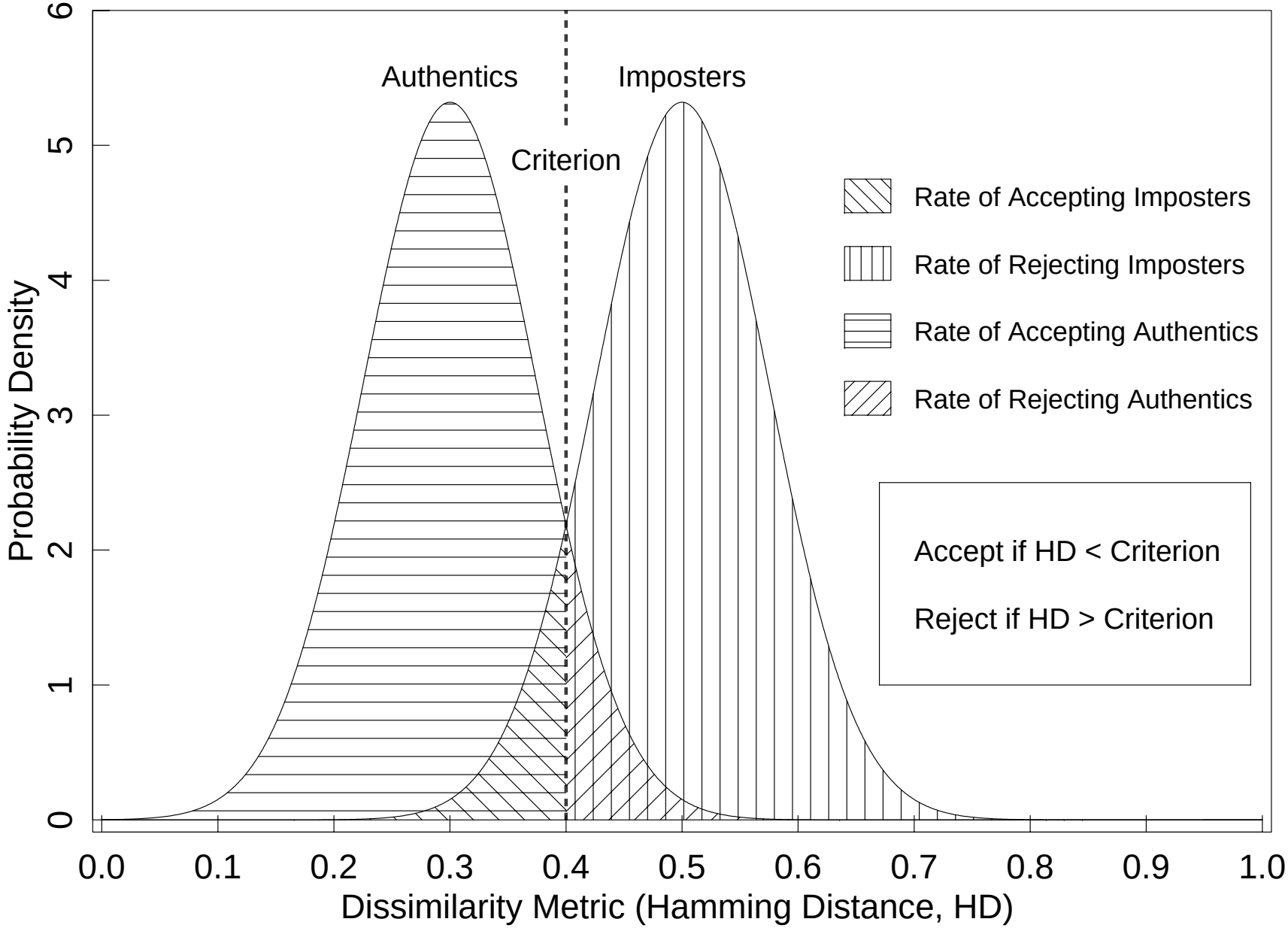
The task is to decide whether or not this feature vector is consistent with a particular class or object category. Thus the problem of classification amounts to a "same / different" decision about the presenting feature vector, compared with vectors characteristic of certain object classes.

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

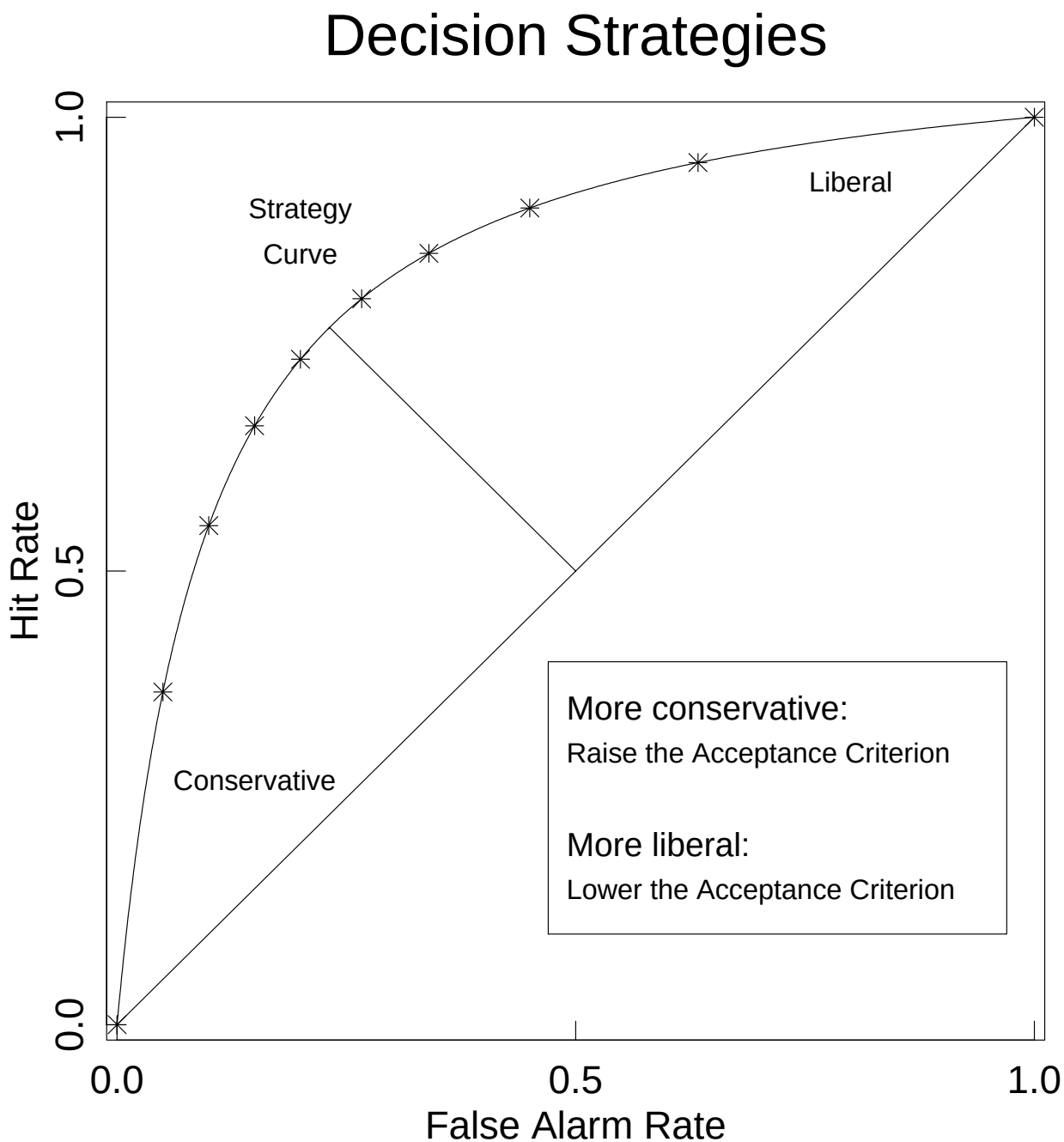
Statistical Decision Theory



(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):

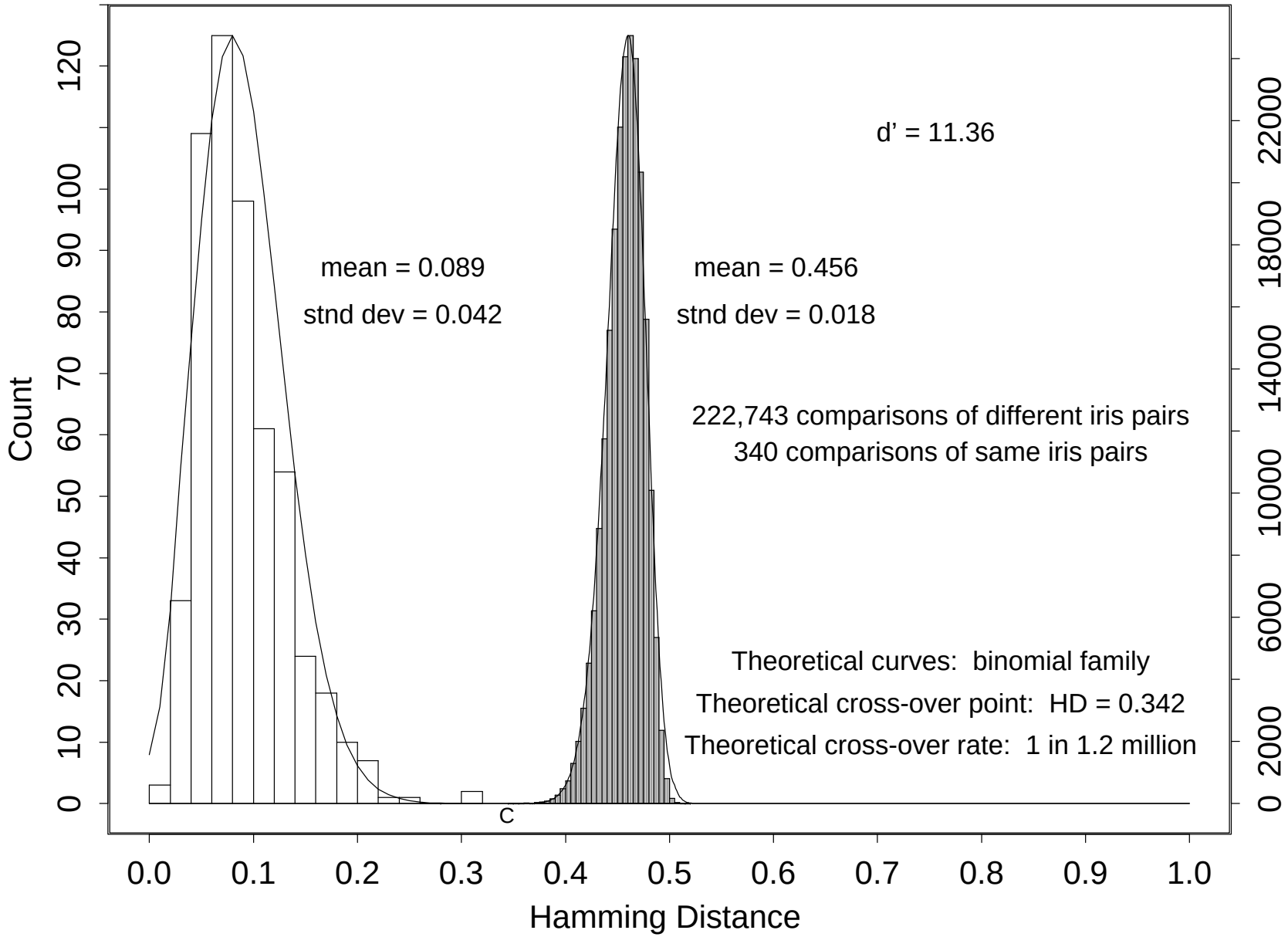
$$d' = \frac{|\mu_2 - \mu_1|}{\sqrt{\frac{1}{2}(\sigma_2^2 + \sigma_1^2)}}$$

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

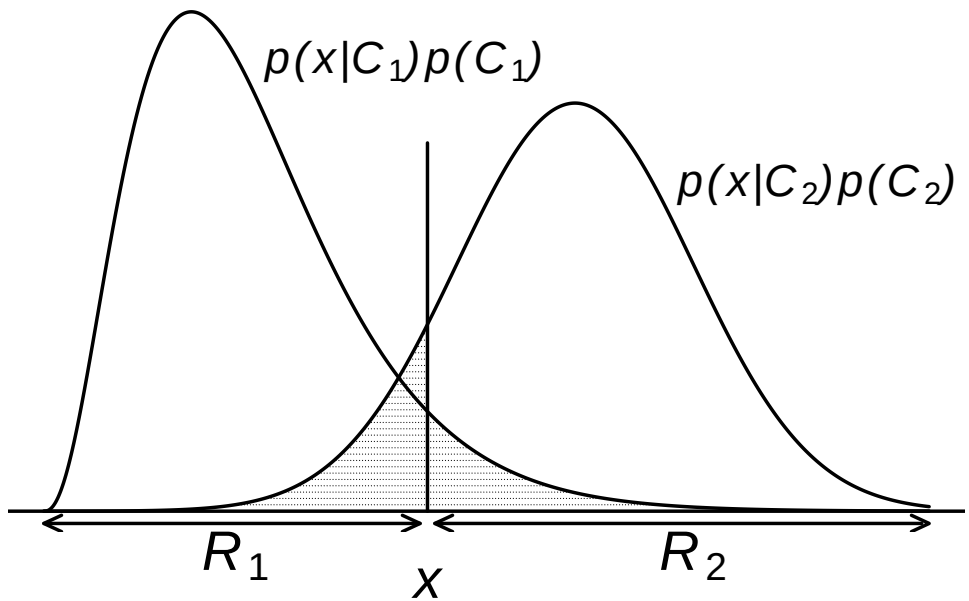


(Statistical Decision Theory, con't)

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 $d' \approx 11$.

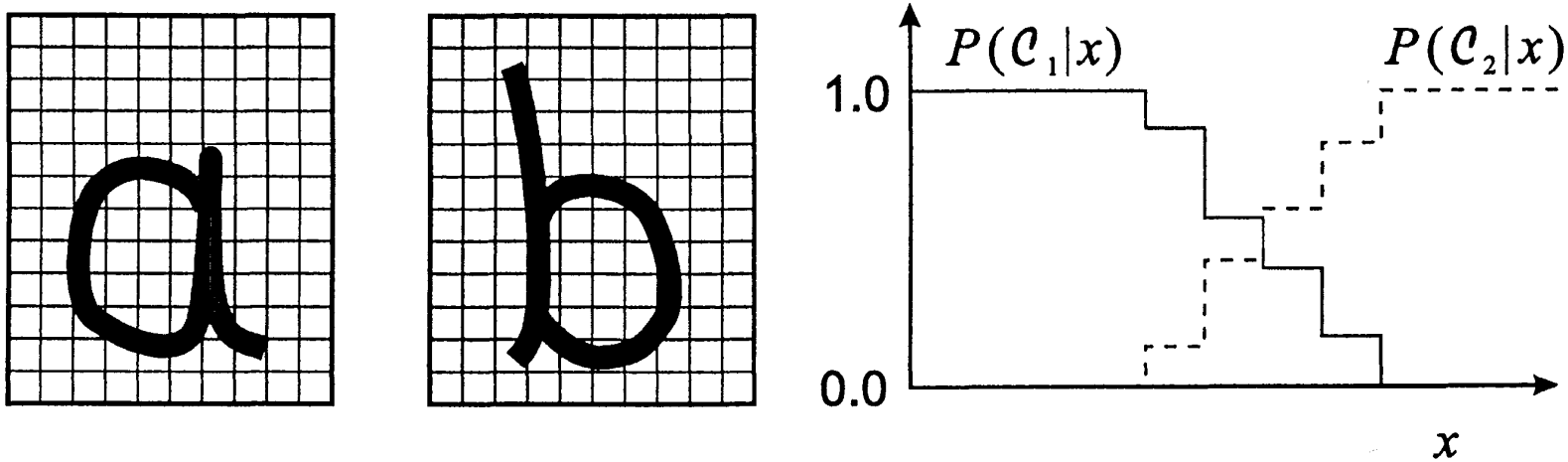
Because reliability of pattern recognition depends on the between-class variance being larger than the within-class variance, R. Fisher defined the "separation between two distributions" as the ratio of their between-class variance to their within-class variance. This definition is related to d'.

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:

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

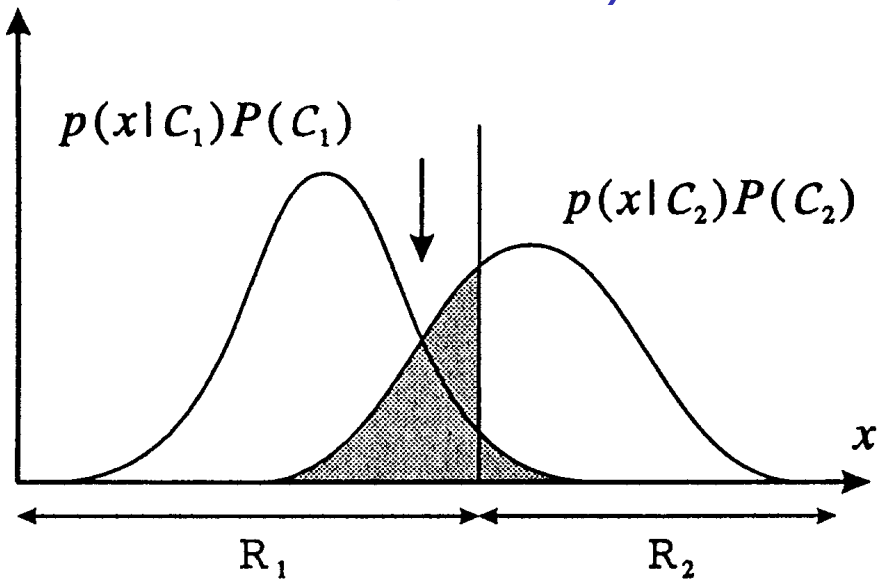
Since the denominator $P(x)$ in Bayes' Rule is independent of C_k, we can rewrite this minimum misclassification criterion simply as:

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:

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

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:

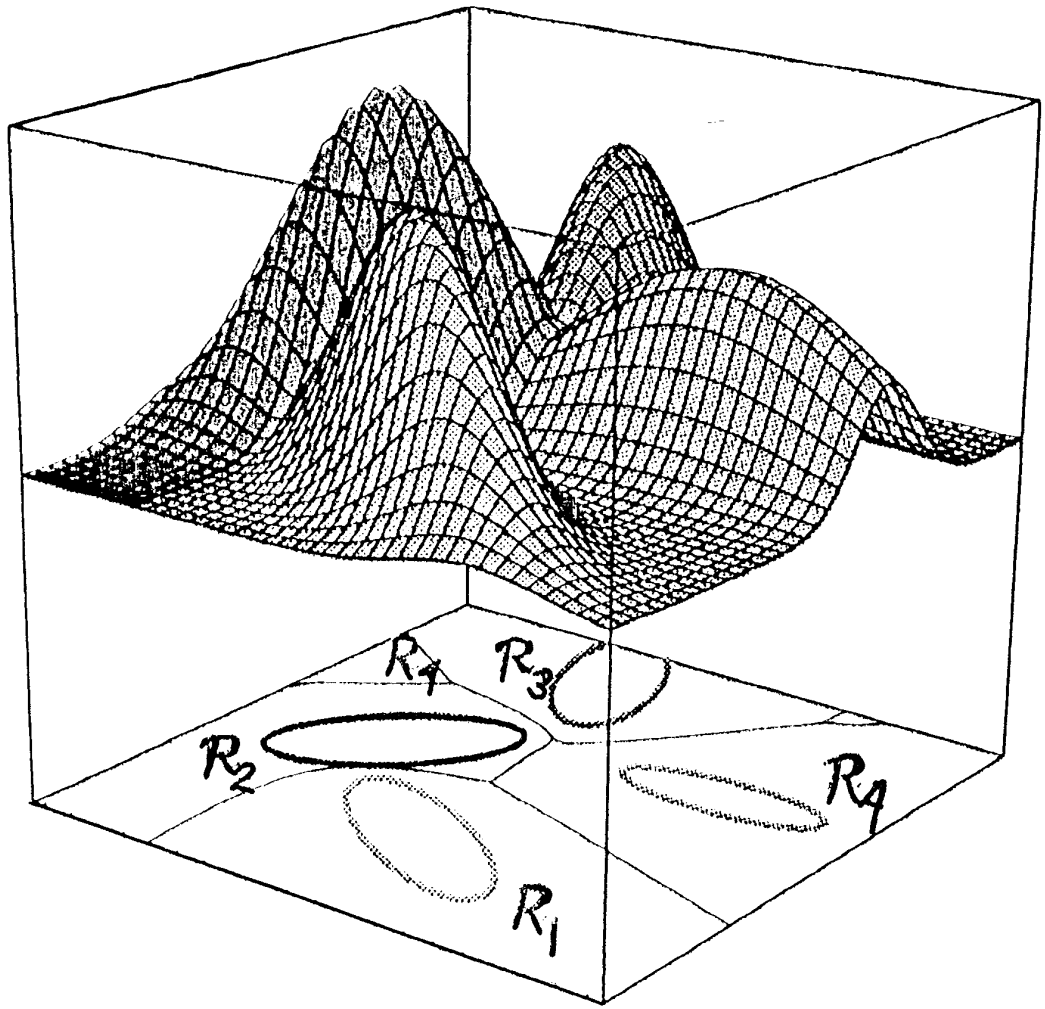
$$y_k(x) = P(C_k|x)$$

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.)



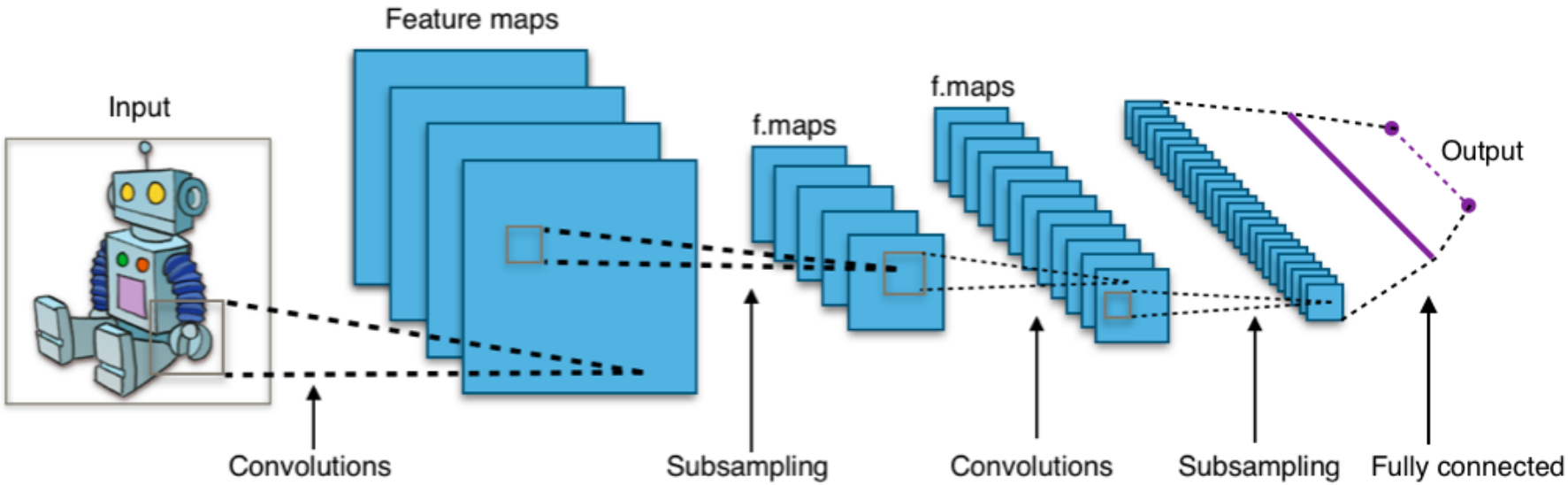
Discriminative versus Generative Methods in Vision

- 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)

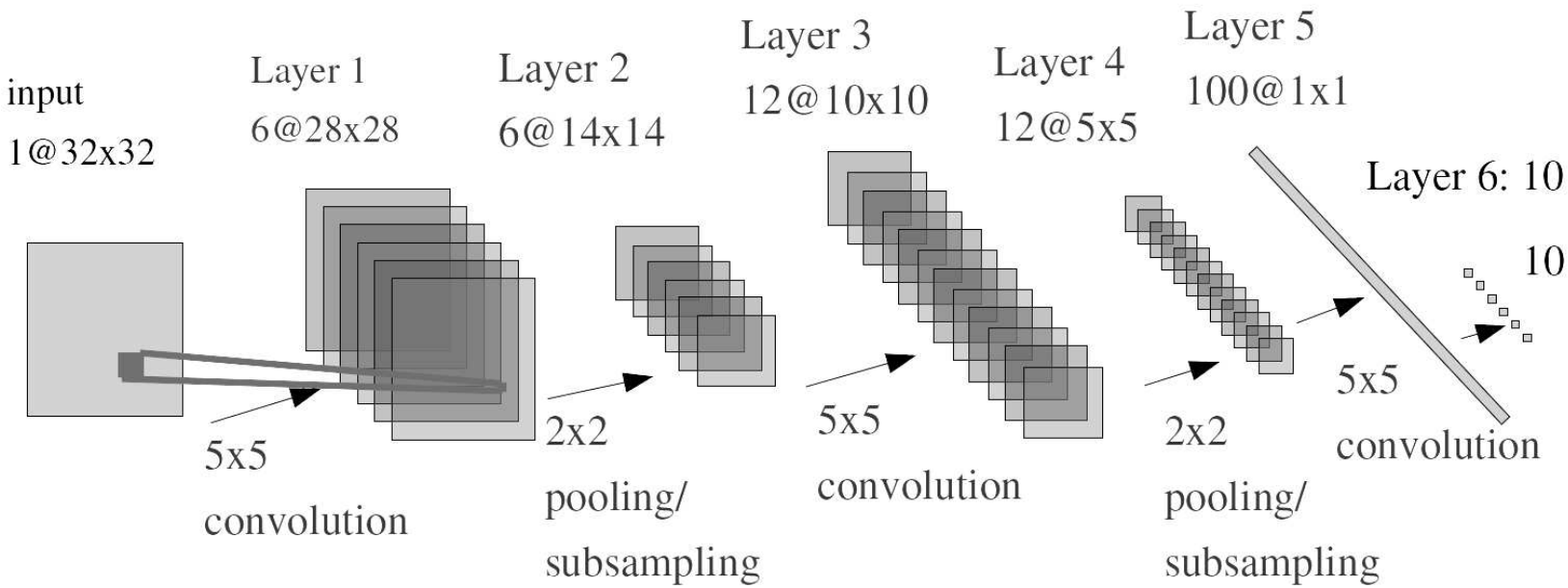


Example: Convolutional Neural Network for OCR (LeCun)

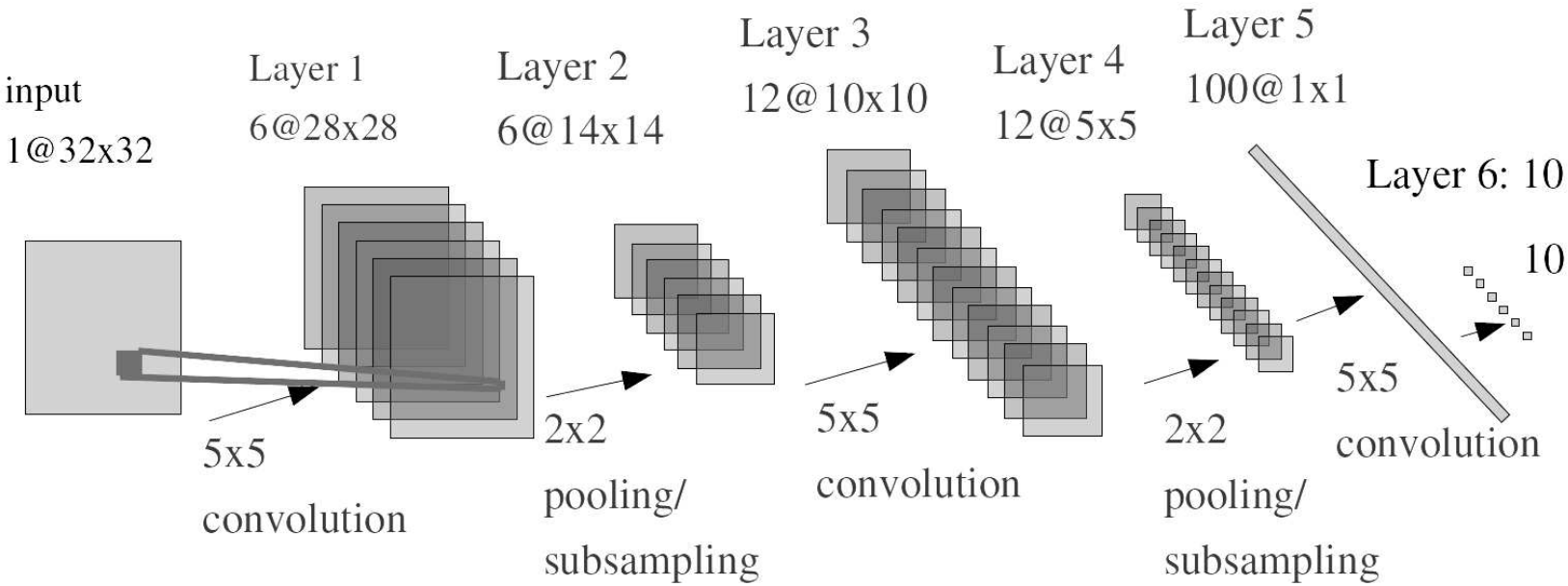
Optical Character Recognition systems have many applications:

- 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)



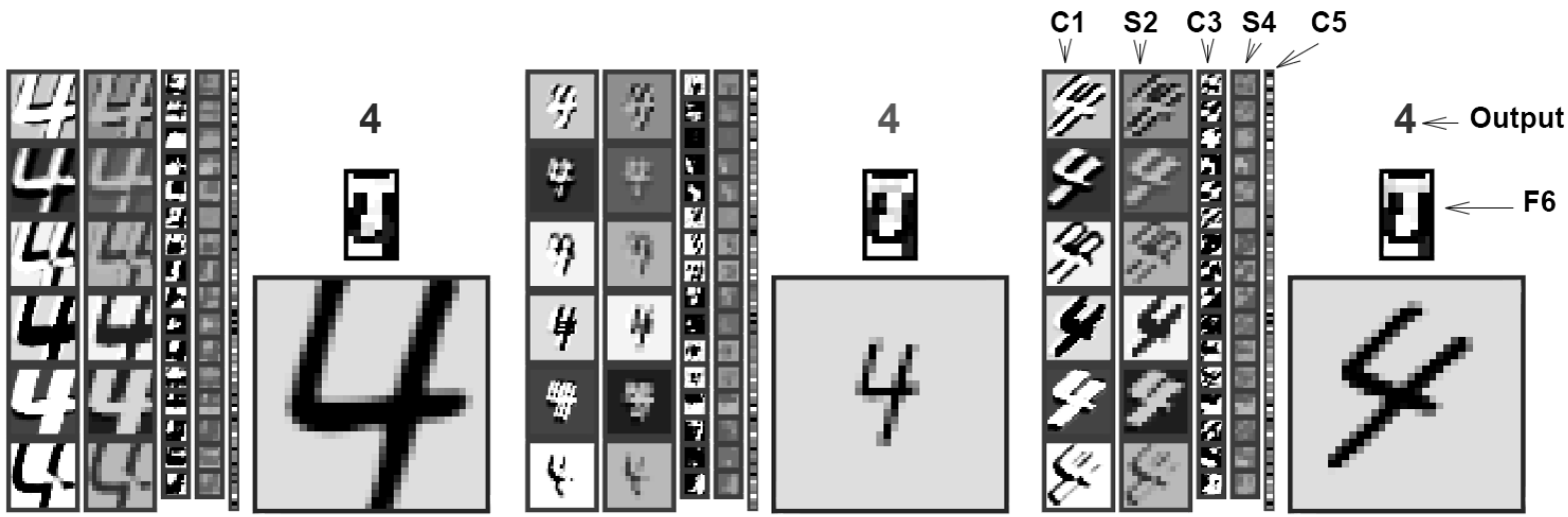
- Input is a 32×32 pixel image, containing some digit or character
- In the training phase, 100,000s of examples of each target are used
- Training is supervised back-propagation: target output is set to $+1$, all others to -1. Errors back-propagate to adaptable feature maps
- Neurons in a feature map 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/`

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; full siblings; double cousins) sometimes look almost identical (apart from age cues):

(Face Detection, Recognition, and Interpretation, con't)

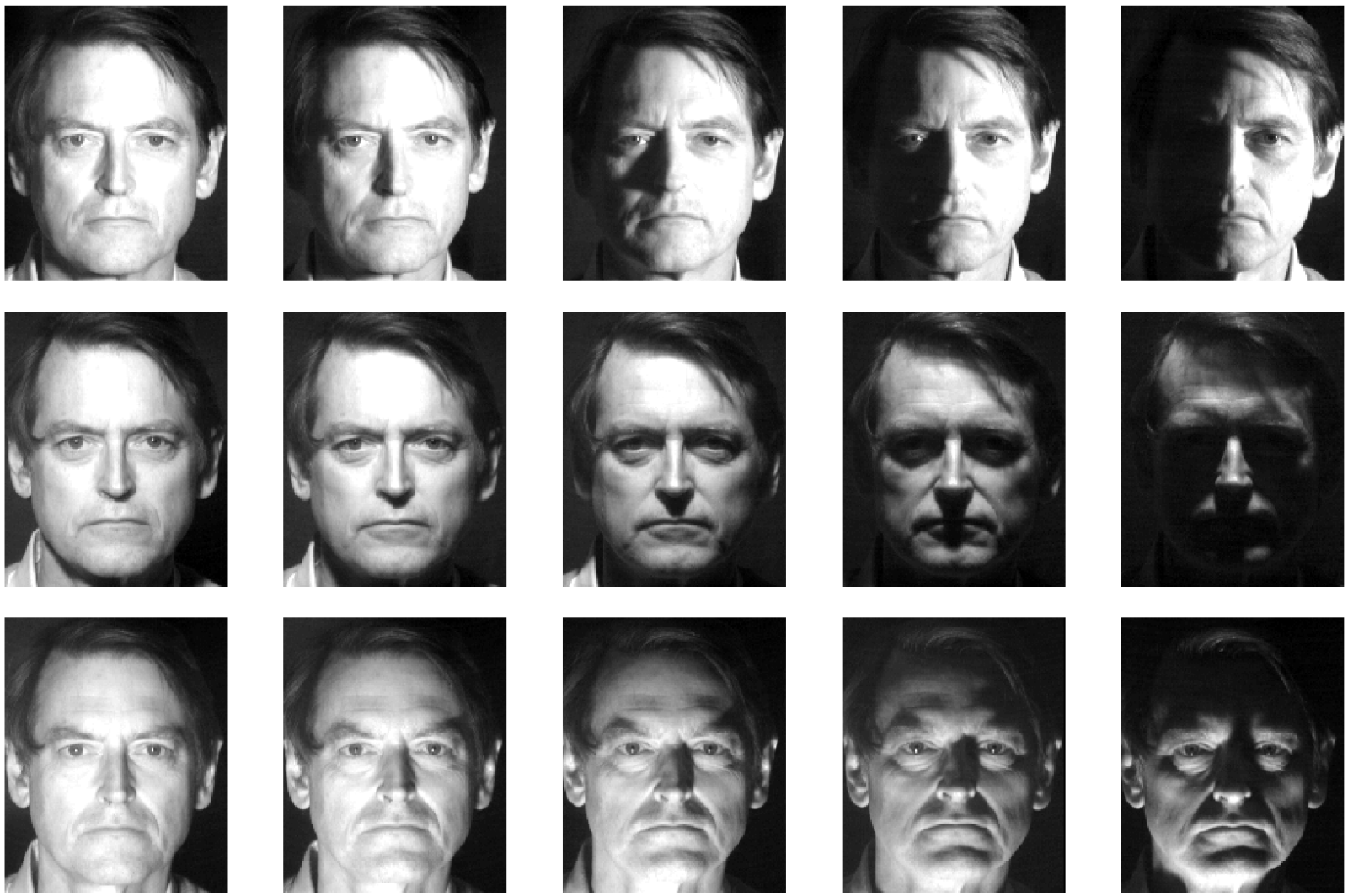
Classic problem: within-class variation (same person, different conditions) can exceed the between-class variation (different persons).

...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)

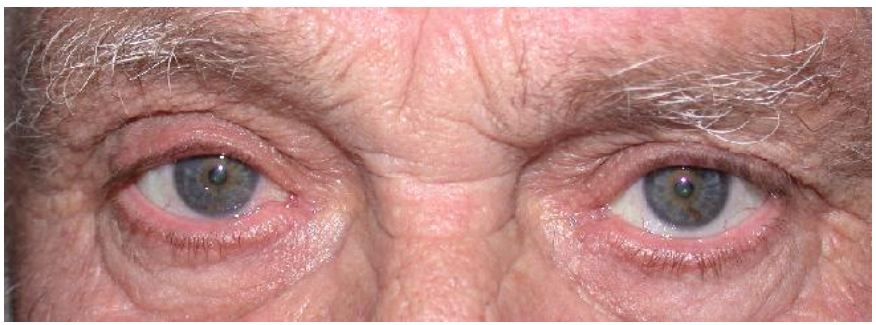
Classic problem: within-class variation (same person, different conditions) can exceed the between-class variation (different persons).

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

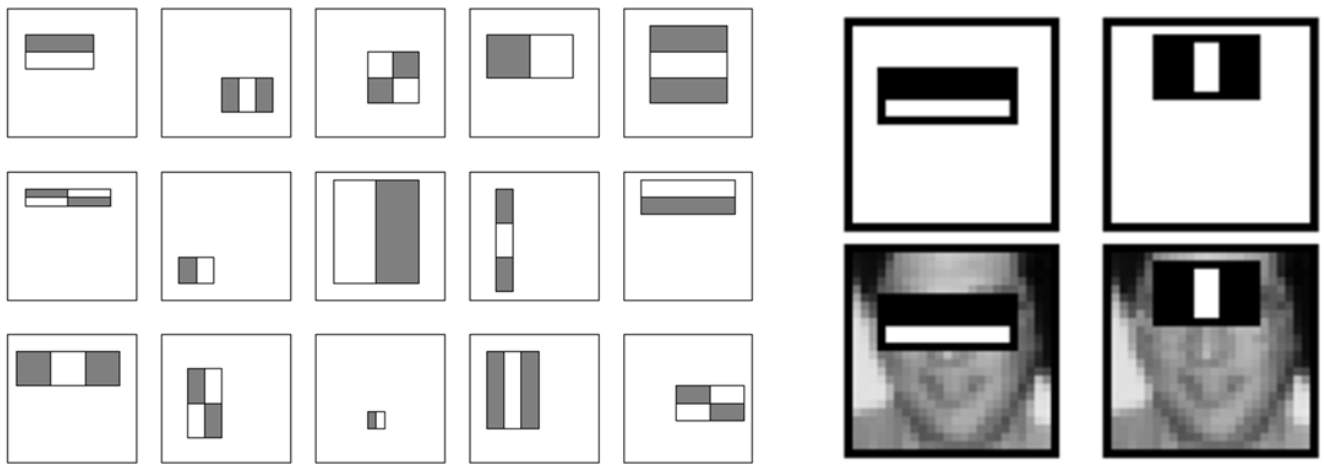
(Face Detection, Recognition, and Interpretation, con't)

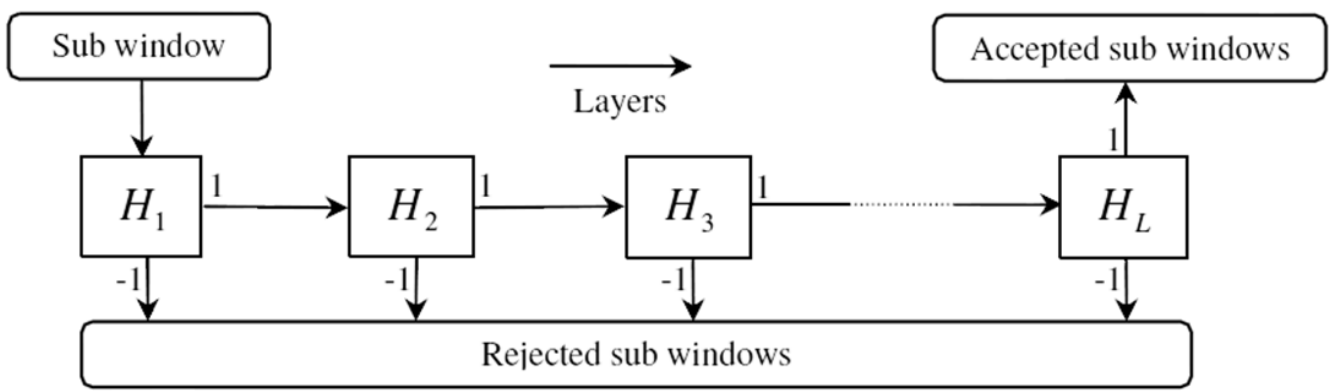
Detecting and recognising faces raises all the usual questions encountered in other domains of computer vision:

- 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

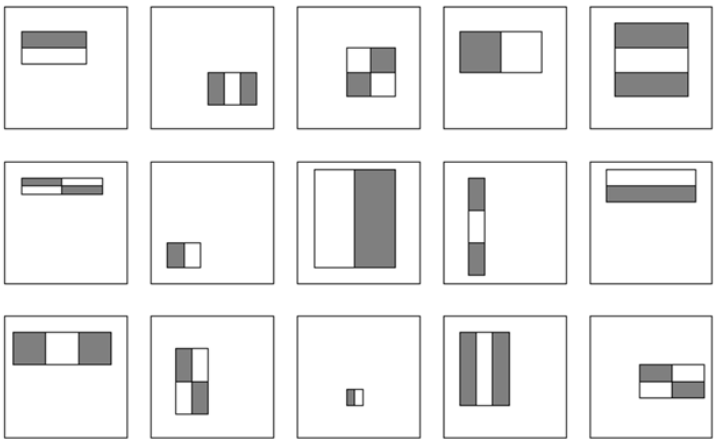
$$h_j(x) = \begin{cases} -p_j & \text{if } f_j < \theta_j \\ p_j & \text{otherwise} \end{cases}$$

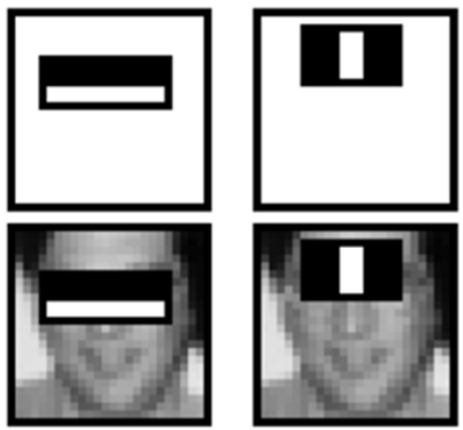
- A strong classifier $h(x)$ takes a linear combination of weak classifiers, using weights α_j learned in a training phase, and considers its sign:

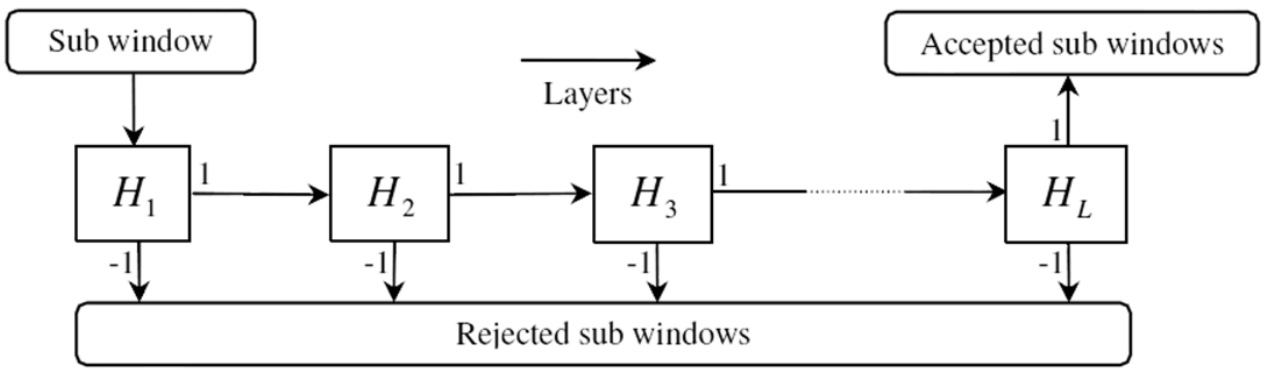
$$h(x) = \text{sign}(\sum_j \alpha_j h_j)$$

(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)

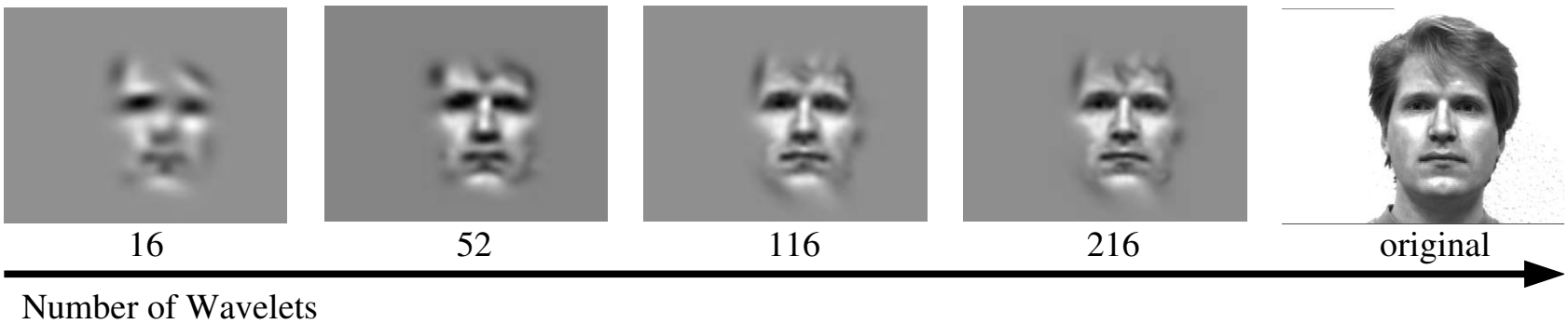
- Training uses the AdaBoost ("Adaptive Boosting") algorithm
- This supervised machine learning process adapts the weights α_j such that early cascade layers have very high true accept rates, say 99.8% (as *all* must detect a face; hence high false positive rates, say 68%)
- Later stages in the cascade, 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:

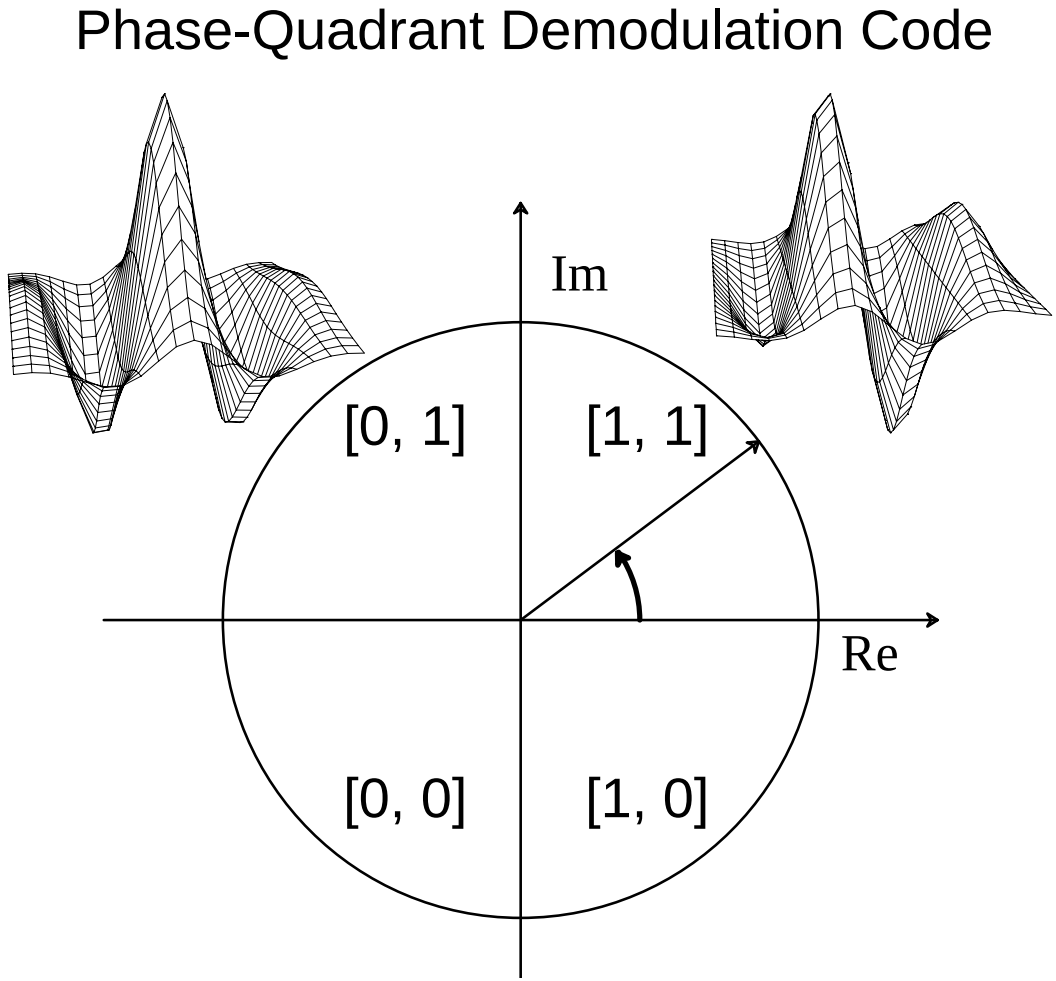
2D Appearance-based Face Recognition: Gabor Wavelets

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)



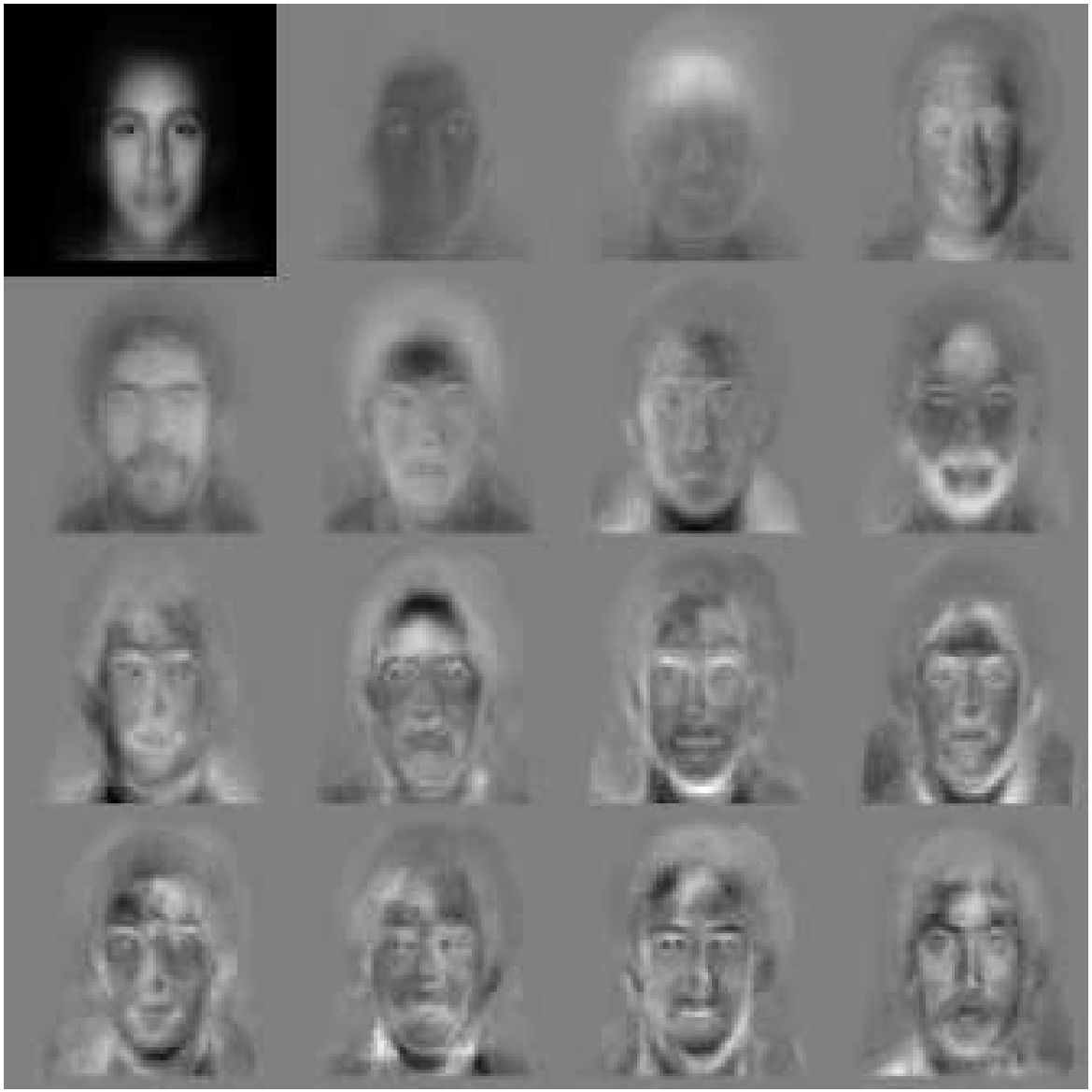
Computed feature vectors in a face code can be local 2D Gabor wavelet amplitude or phase information. Bits in the "face code" are set by the quadrant in which the phasor lies, for each aspect of facial structure.

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

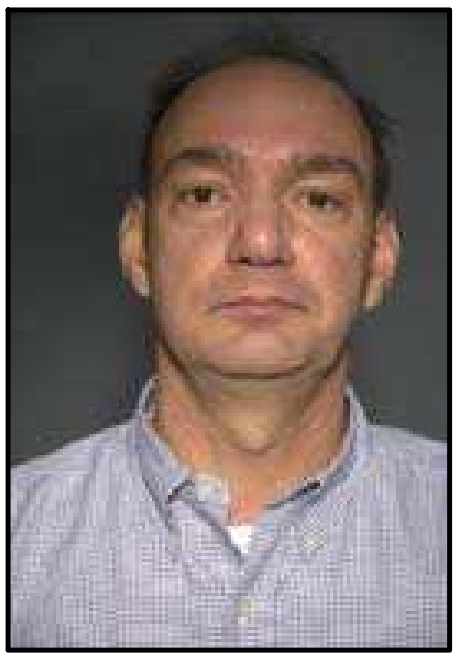
(2D Appearance-based Face Recognition: "Eigenfaces")

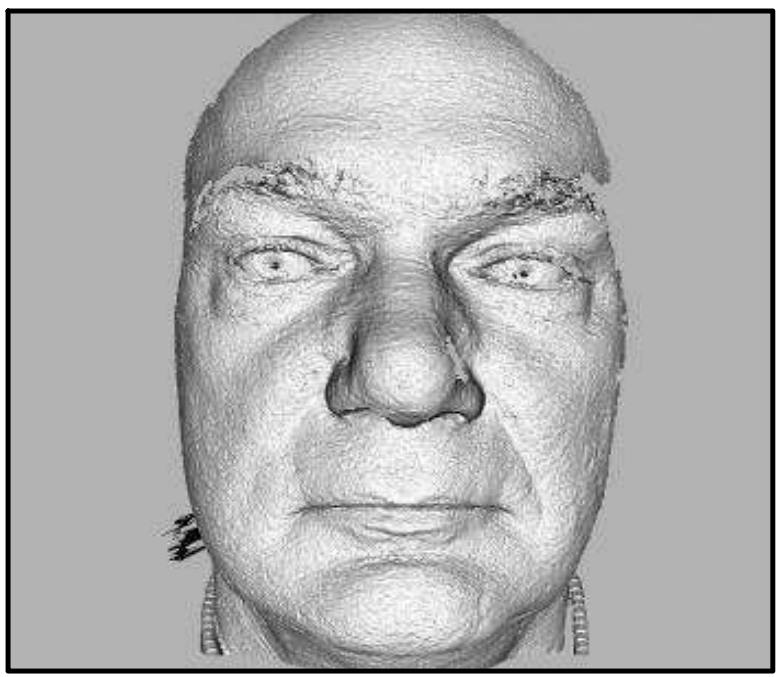
- Performance is often in the range of 90% to 95% accuracy
- Databases can be searched very rapidly, as each face is represented by a very compact feature vector of only about 20 numbers
- A major limitation is that significant (early, low-order) eigenfaces emerging from the statistical analysis arise 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

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:

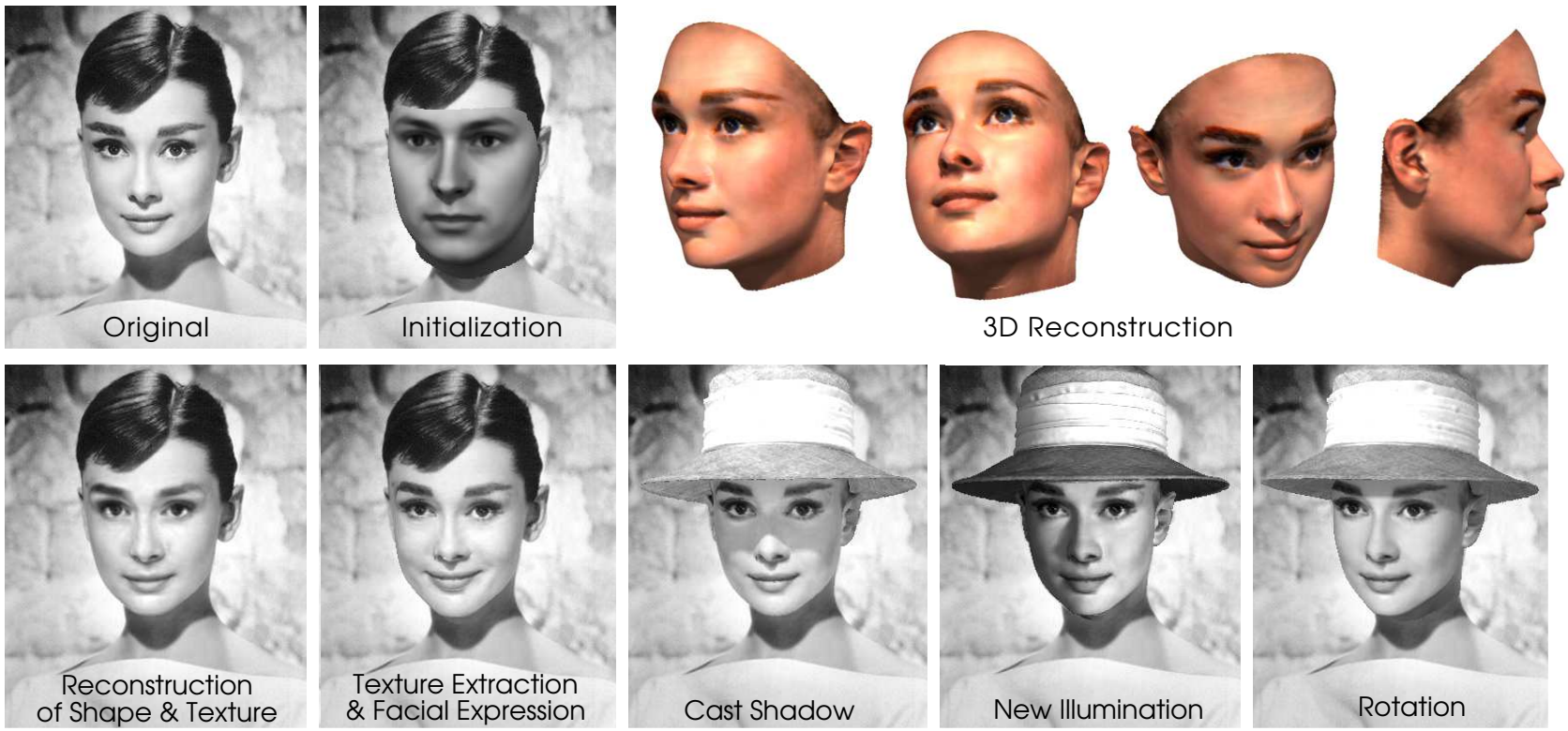
- laser range-finding, even down to millimetre resolution
- calibrated stereo cameras
- projection of structured IR light (grid patterns whose distortions reveal shape, as with Kinect)
- extrapolation from multiple images taken from different angles

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`

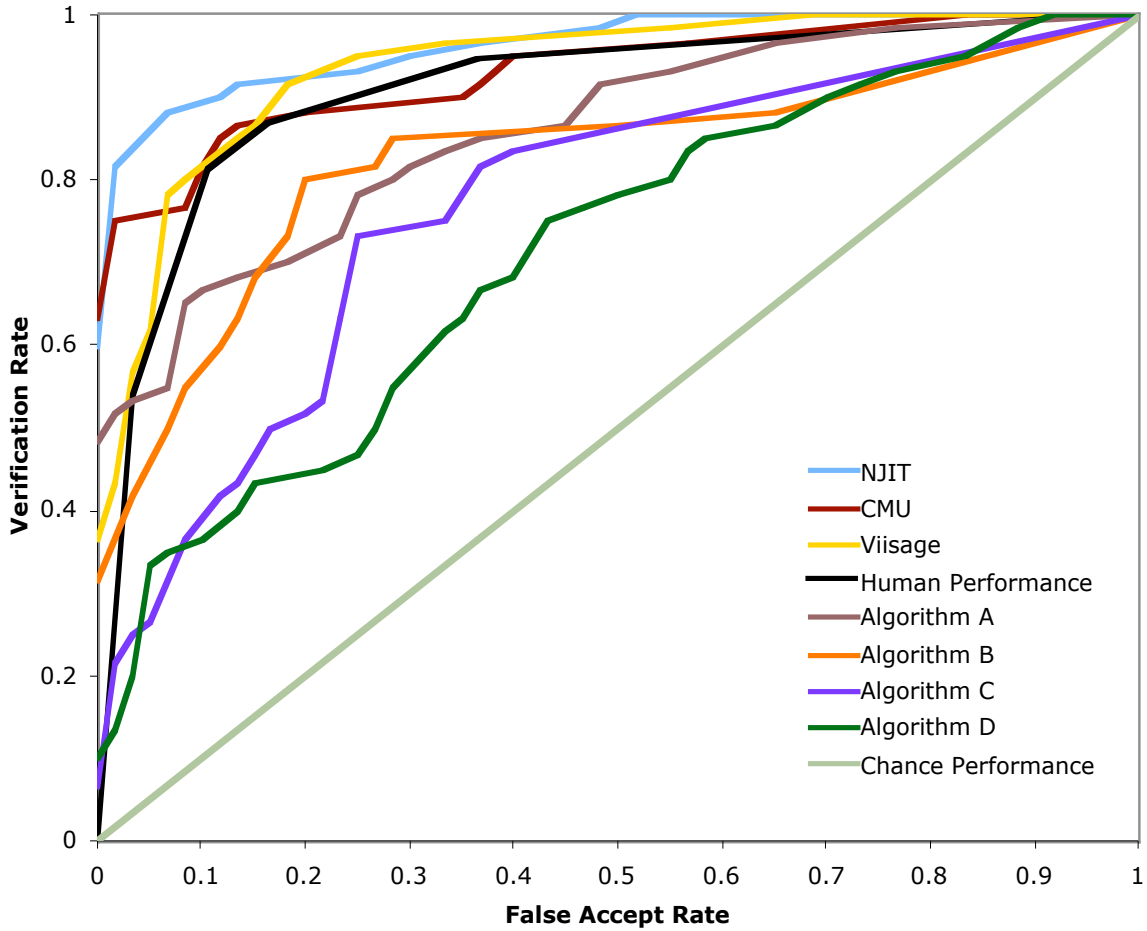
(Three-Dimensional Approaches to Face Recognition)

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. But in a 2007 test, three algorithms had ROC curves above (better than) human performance at non-familiar face recognition (the black curve):



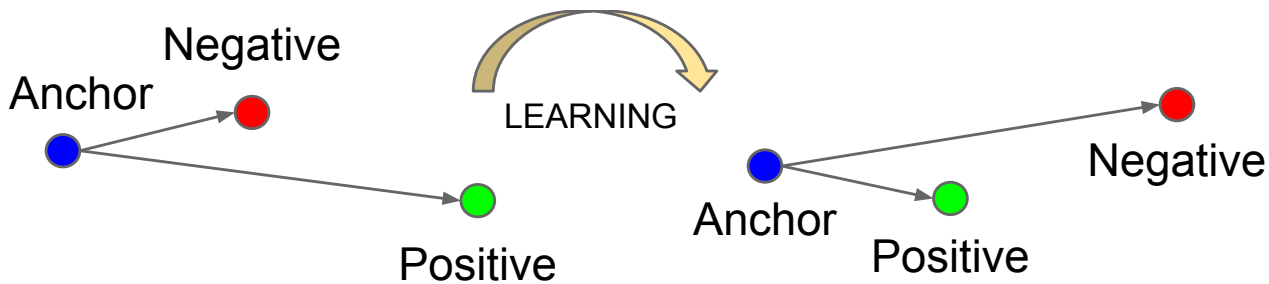
Major Breakthrough in 2015: Deep-Learning "FaceNet"

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 are new accuracy records.

(Major Breakthrough in 2015: Deep-Learning "FaceNet")

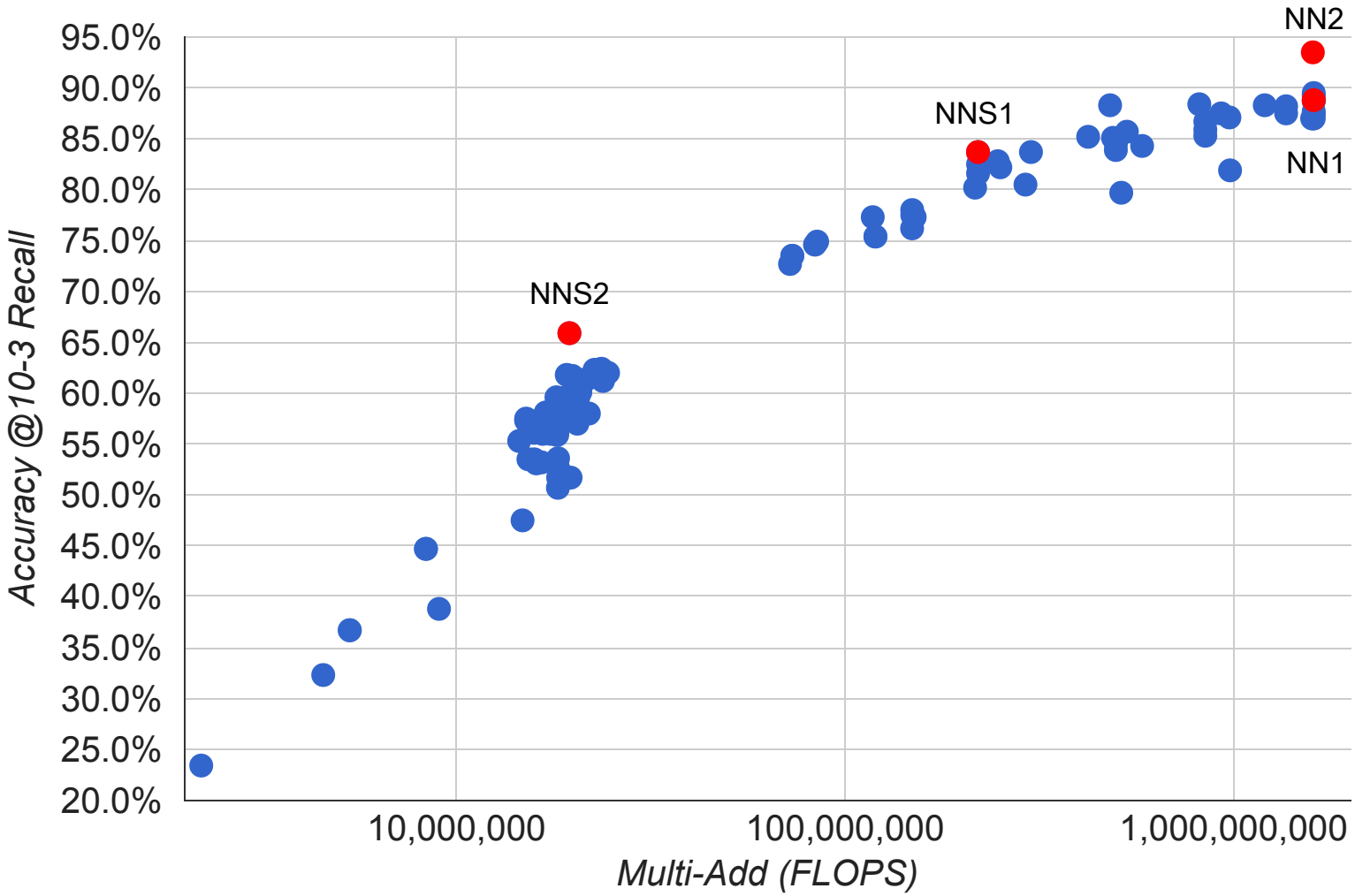
- Convolutional Neural Net with 22 layers and 140 million parameters
- Big dataset: trained on 200 million face images, 8 million identities
- 2,000 hours training (clusters); about 1.6 billion FLOPS per image
- Euclidean distance metric (L2 norm) on embeddings $f(x_i)$ learned for cropped, but not pre-segmented, 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

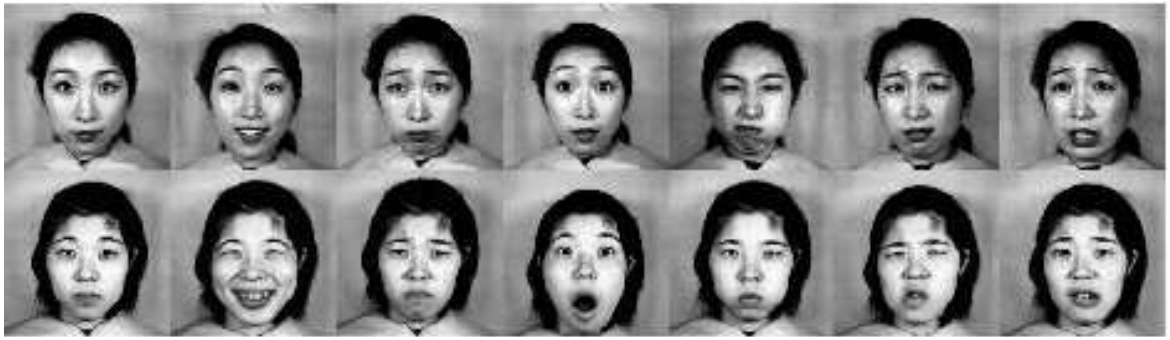
(Major Breakthrough in 2015: Deep-Learning "FaceNet")



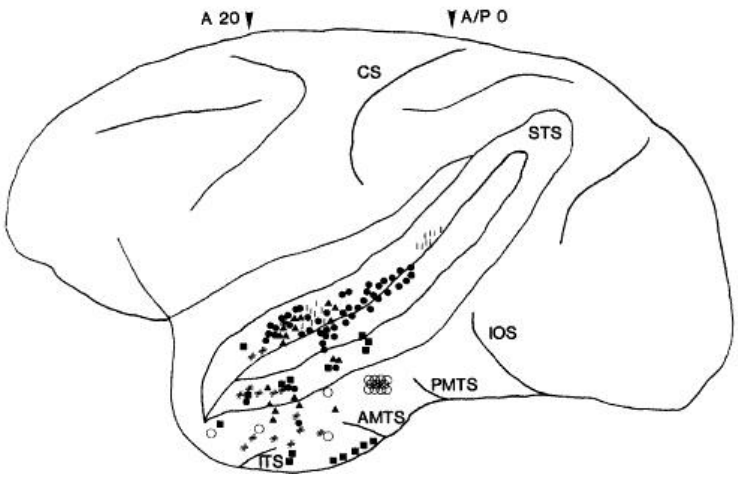
Different variants of the Convolutional Neural Net and model sizes were generated and run, revealing the trade-off between FLOPS and accuracy for a particular point on the ROC curve (False Accept Rate = 0.001)

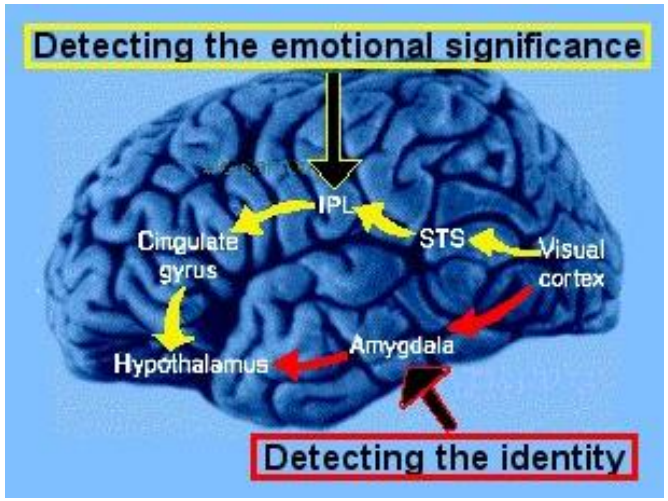
Affective Computing: Interpreting Facial Emotion

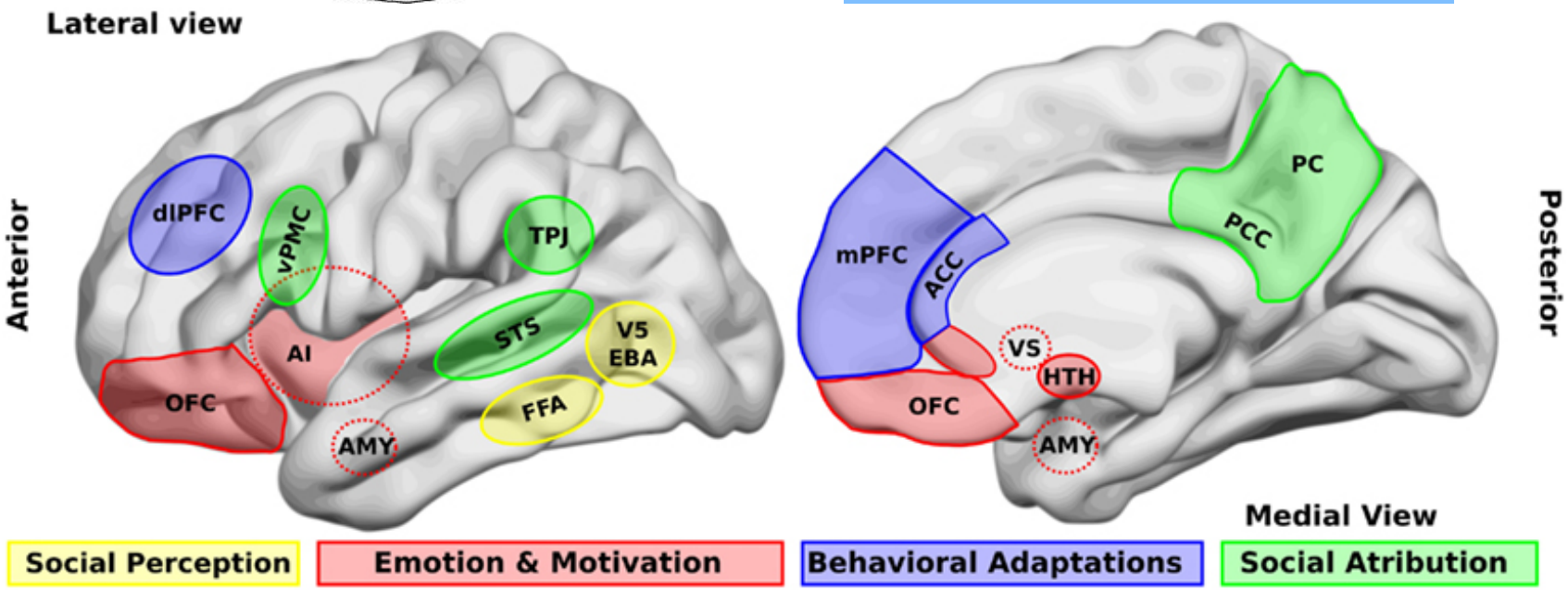
Humans use their faces as visually expressive organs, cross-culturally



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



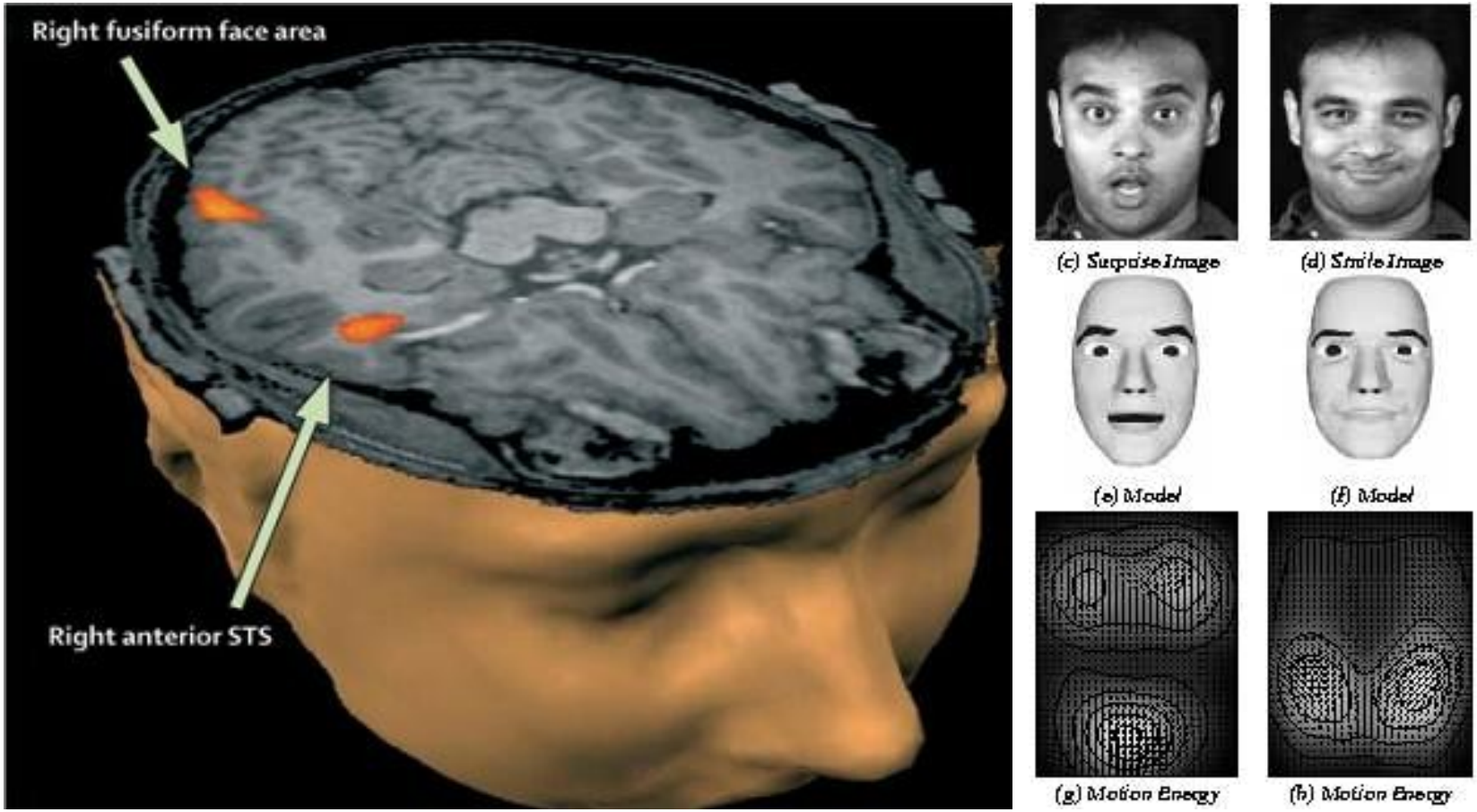




Affective Computing: Classifying Identity *and* Emotion

- 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.