(3rd Supplementary Slides: Computer Vision)

Professor John Daugman

University of Cambridge

Computer Science Tripos, Part II
Lent Term 2015/16

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.

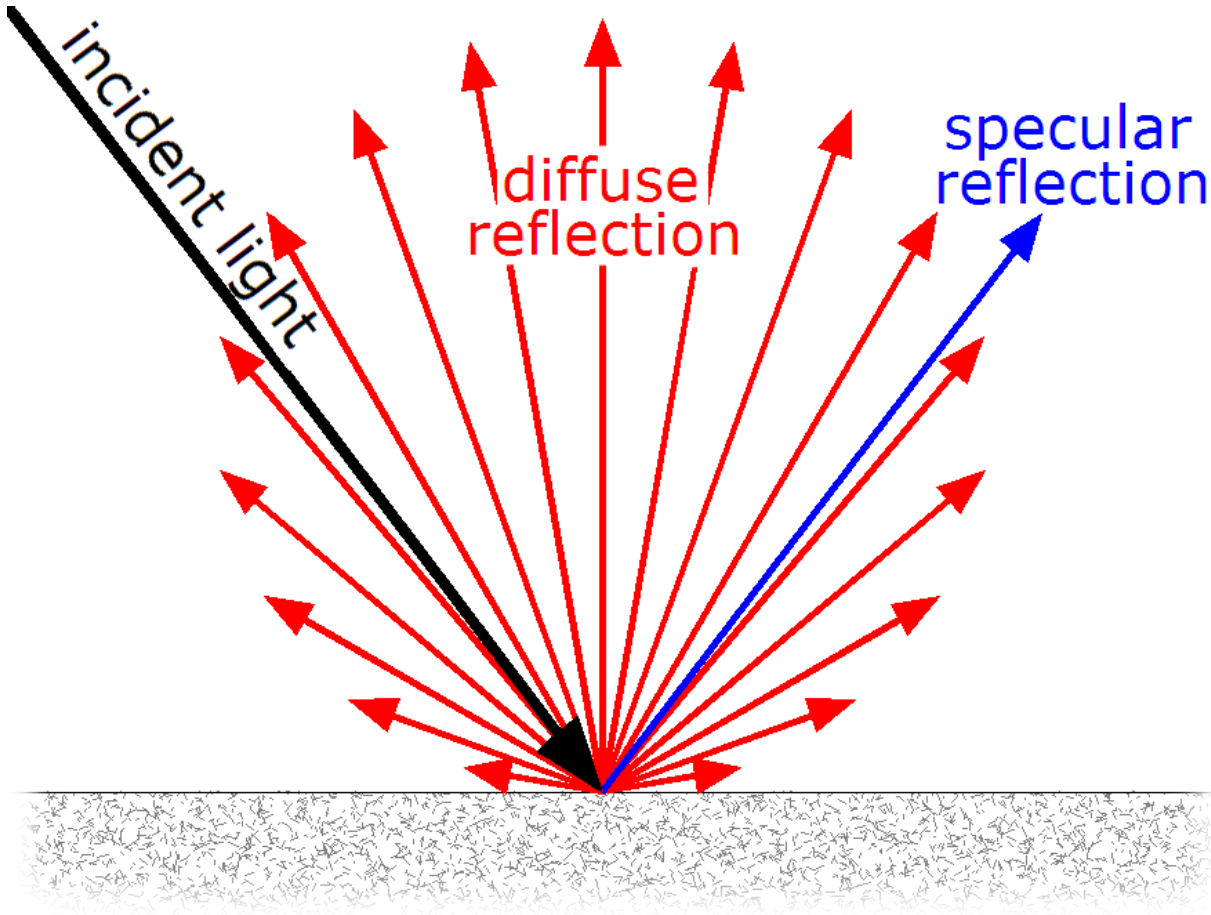
- Surface albedo is the fraction of the illuminant that is re-emitted from a surface in all directions. Thus it relates to how "light" or "dark" is the surface, and this may vary locally across it
- The amount of light reflected is the product of two factors: the surface albedo, times a geometric factor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

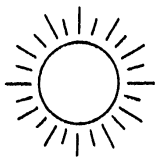


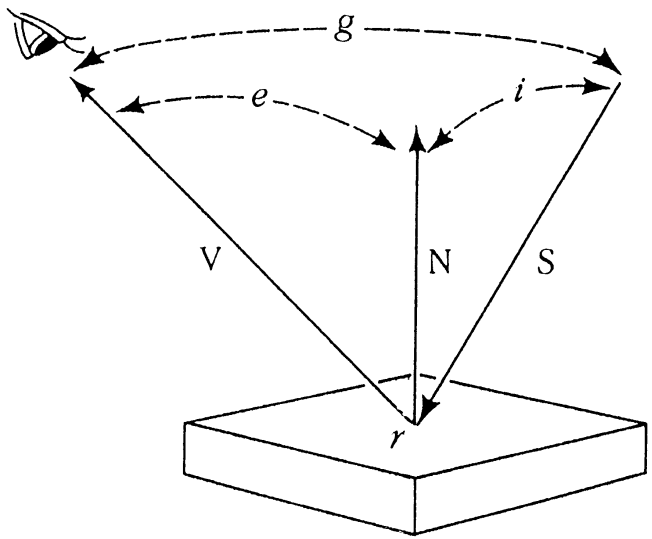
(Surfaces and Reflectance Maps, con't)

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)

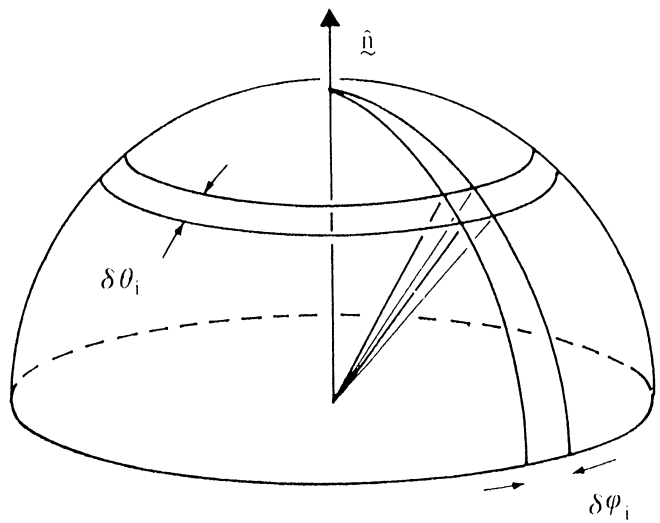
Typically, surfaces have both specular and matte properties. For example, facial skin may vary from Lambertian (powdered) to specular (oily). The purpose of powdering one's face is to specify s and n in this expression:

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

- 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



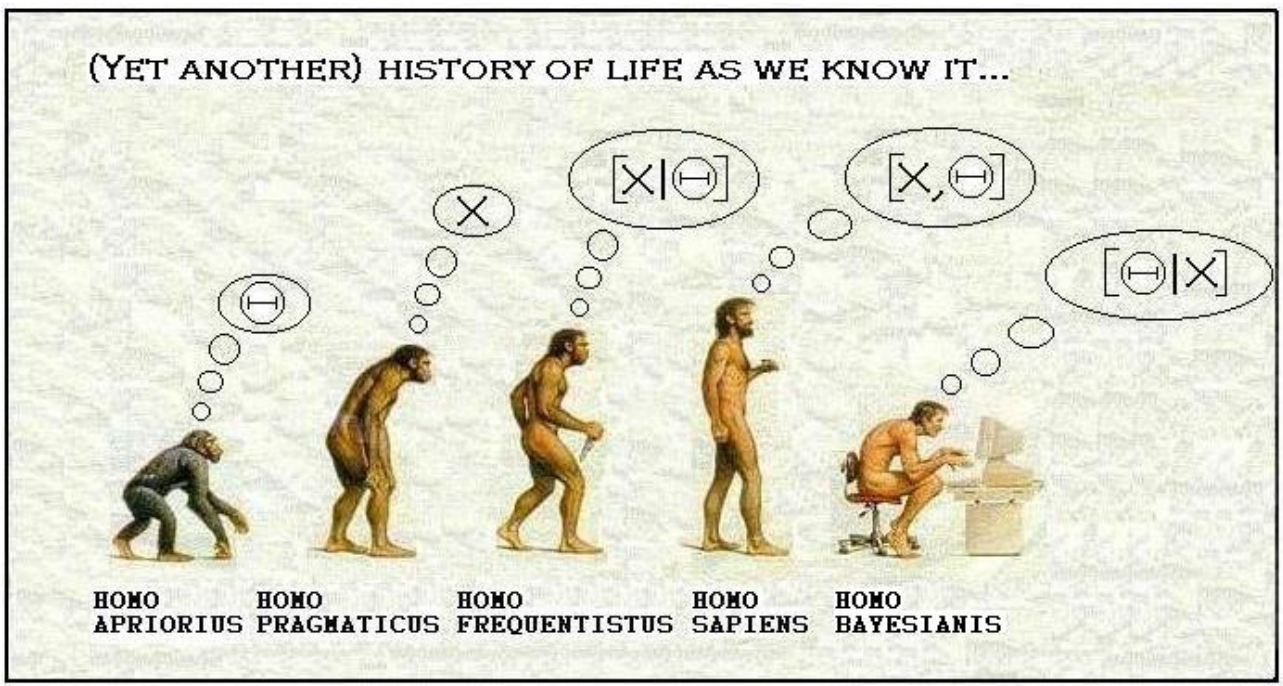
Clearly, the only way to infer the nature and geometry of surface properties from image properties is to build-in certain assumptions about the nature of the surfaces from other kinds of evidence. This requires us to consider the general problem of inference and integration of evidence

(Surfaces and Reflectance Maps, con't)

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

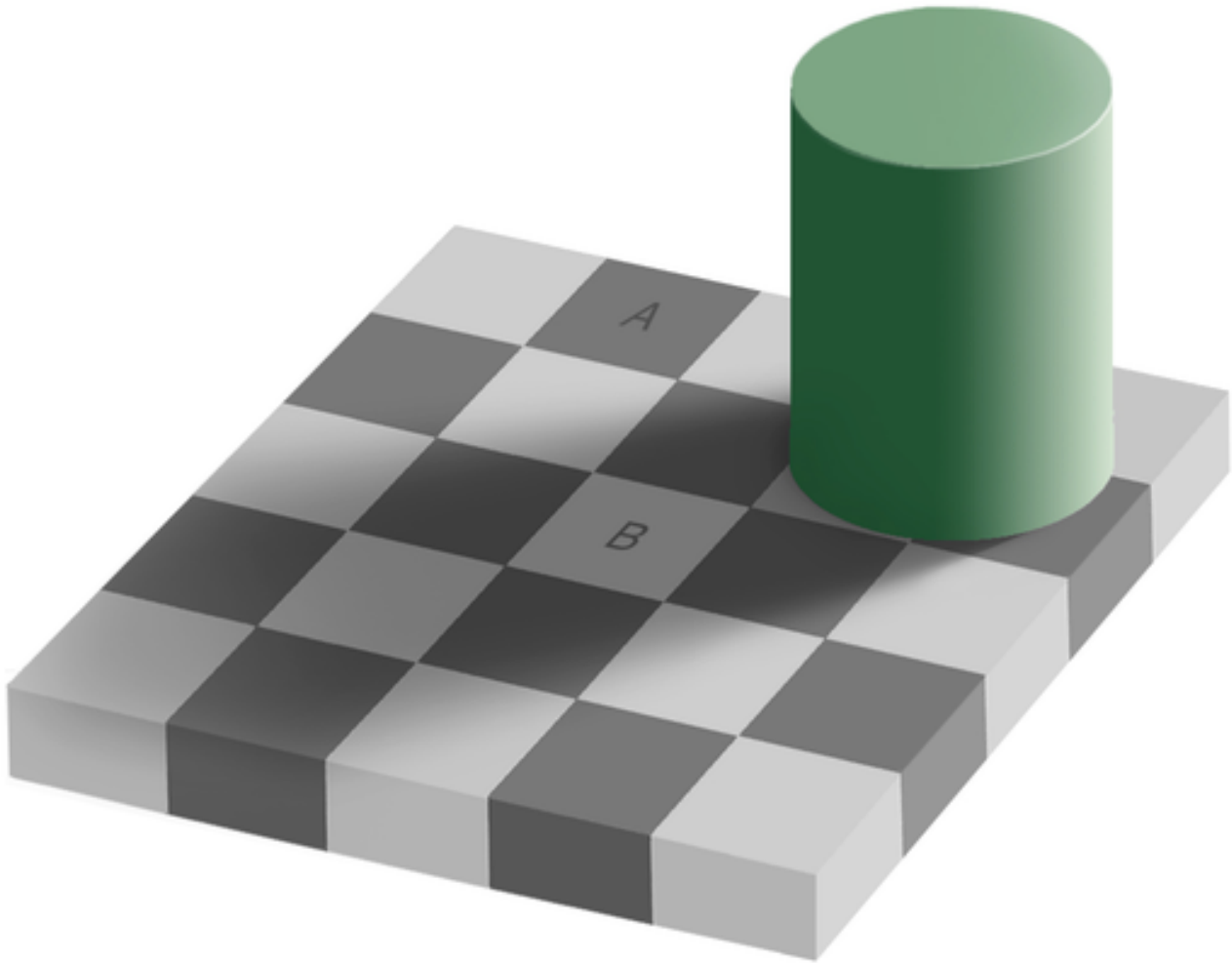
- geometry of the illuminant (e.g. is it a point source, or extended? If a point source, where is it?) Are there several light sources?
- reflectance properties of the surface. What is its reflectance map?
- geometry of the surface (its underlying shape). Are shadows cast?
- rotations of the surface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



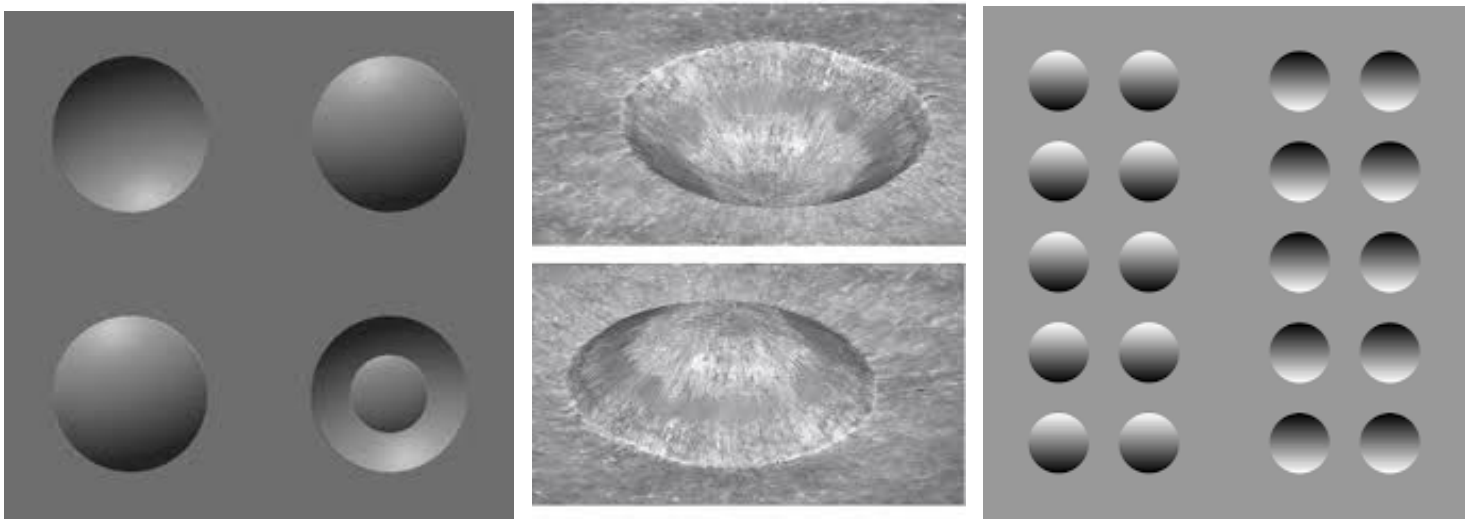
(Surfaces and Reflectance Maps, con't)

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.



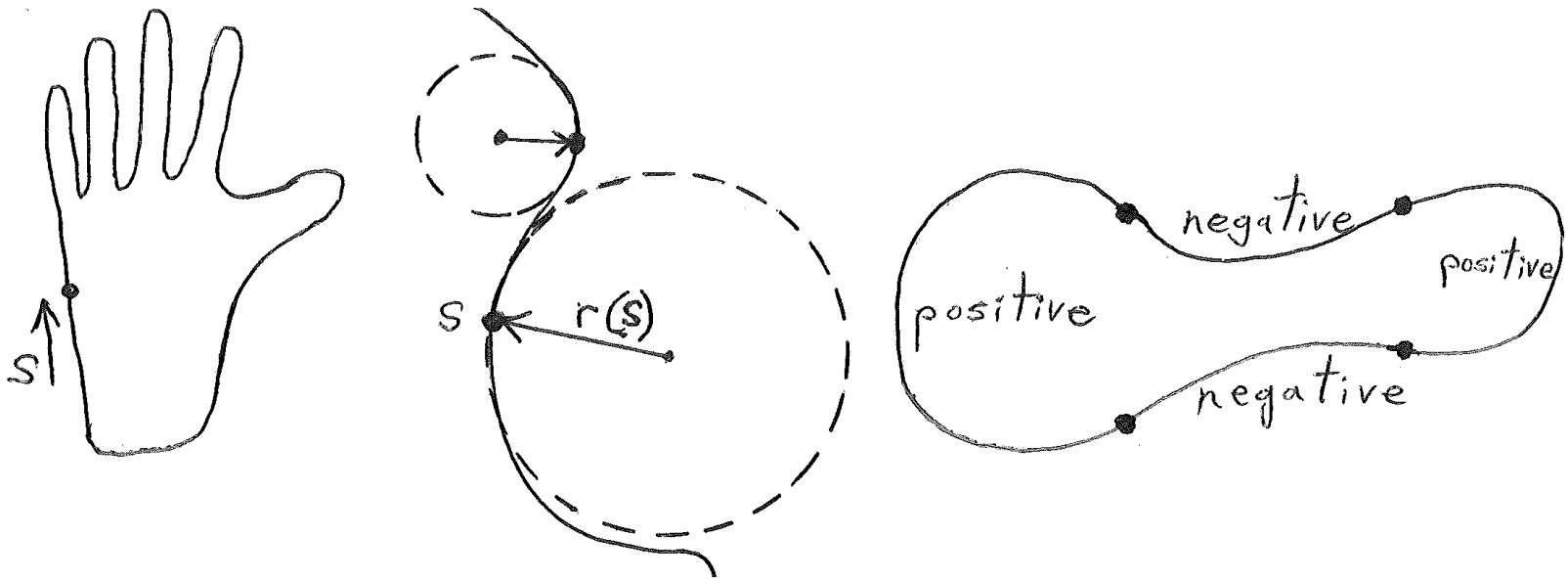
It is almost as though the assumptions (like angle of illumination) are more important than the available image data. The computational nature of the inference task then becomes one of *constraint satisfaction*. Often there are rivalrous (e.g. is it a dome or a crater?) alternative solutions: `http://www.michaelbach.de/ot/fcs_hollow-face/index.html`

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):

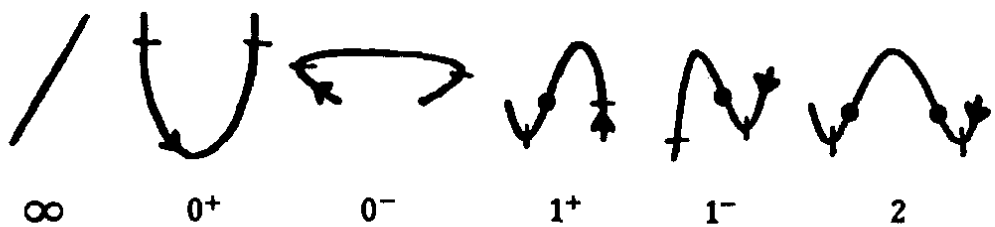
$$\theta(s) \rightarrow K\theta(s)$$

(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).

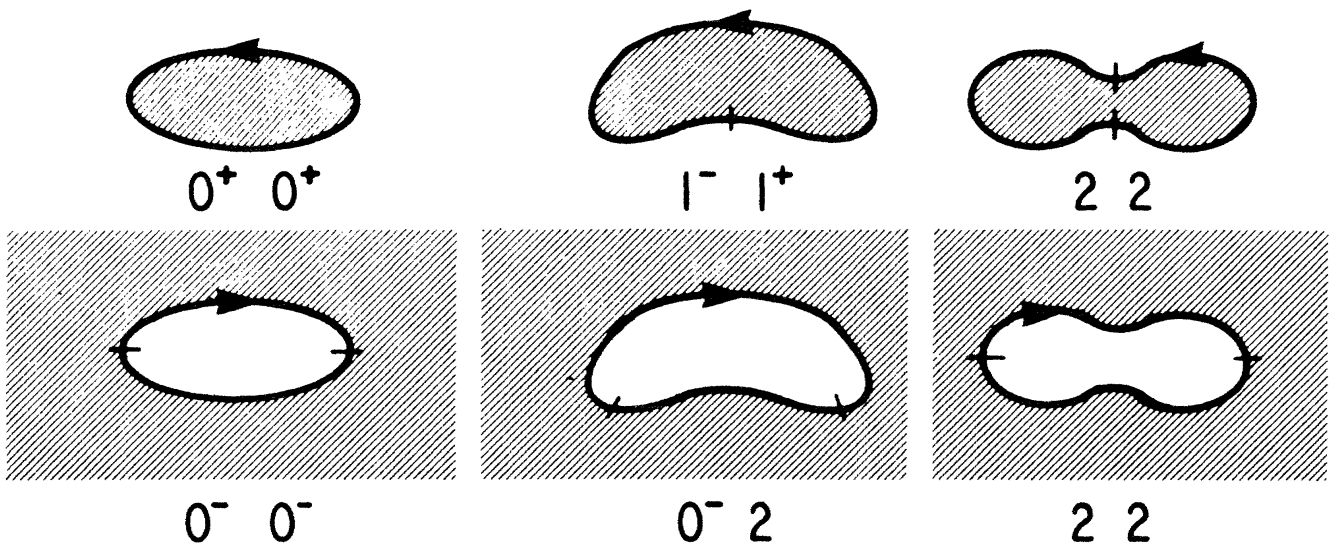
There are just six primitive codon types: all curve segments lying between minima of curvature must have 0, 1 or 2 points of zero curvature, further classified by whether a zero is encountered before ("$-$") or after ("$+$") reaching the maximum curvature in the chosen direction of traversal. Dots show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



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

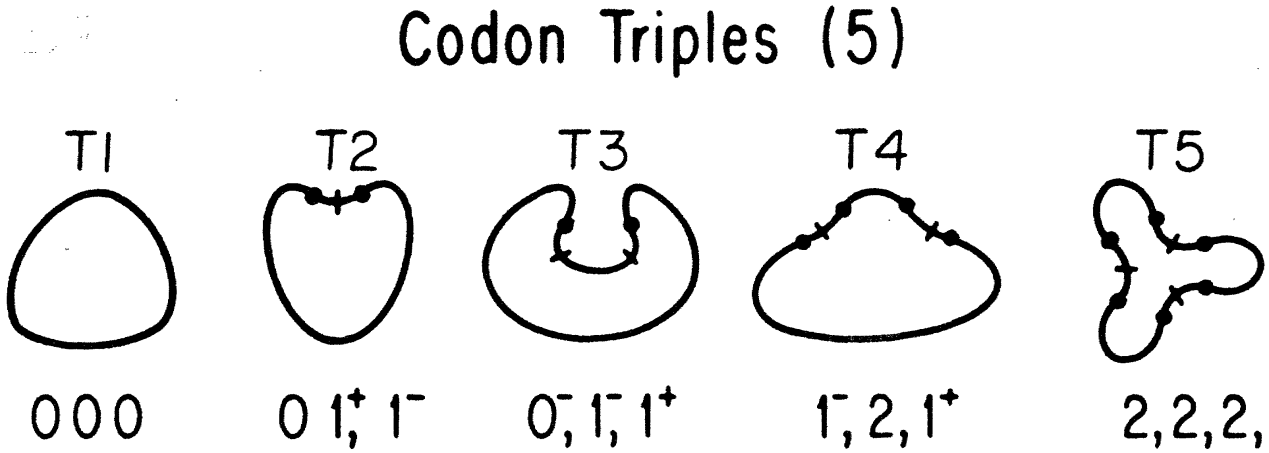
Note that because curvature is a signed quantity, the loci of its minima depend on what we take to be "figure" vs "ground". For open contours like these face profiles (alternatively Rubin's Vase profiles), if we regard "figure" as "to left", then loci of minima depend on direction of traversal:

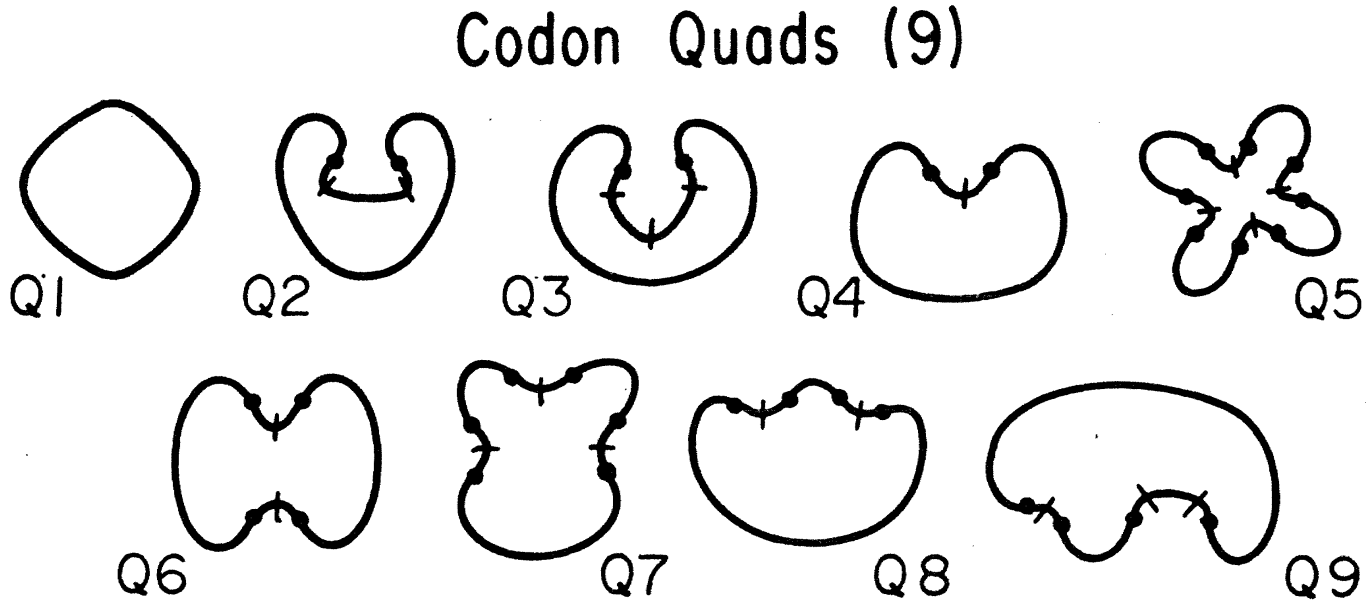
There are 3 possible Codon Pairs (string type depending on direction):



(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.

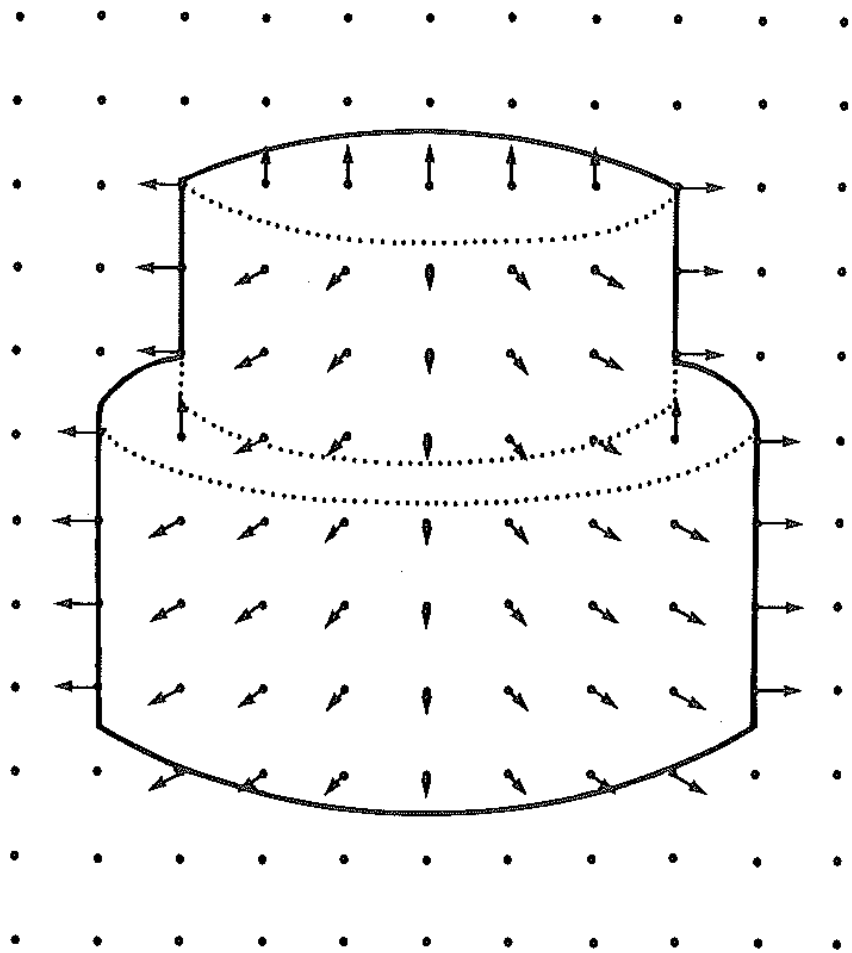
The inflexion (a zero of curvature) of a 3D curve is preserved under 2D projection, thereby guaranteeing that the ordinal relations among the extrema of curvature will also be preserved when projected to an image.

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:

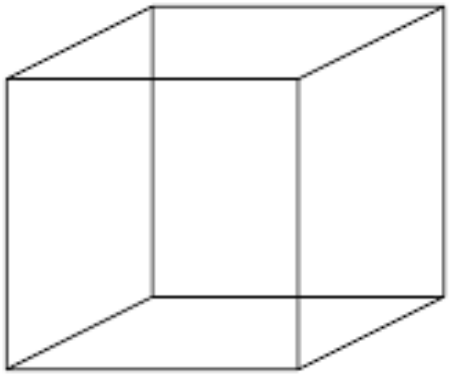
- cylinders: $(\alpha, \beta, \gamma) = (2, 2, 100)$ and $A = B$
- rectangular solids: $(\alpha, \beta, \gamma) = (100, 100, 100)$
- prolate spheroids (shaped like zeppelins): $(\alpha, \beta, \gamma) = (2, 2, 2)$ and (say) $A = B$ but $C < (A, B)$
- oblate spheroids (shaped like tomatoes): $(\alpha,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

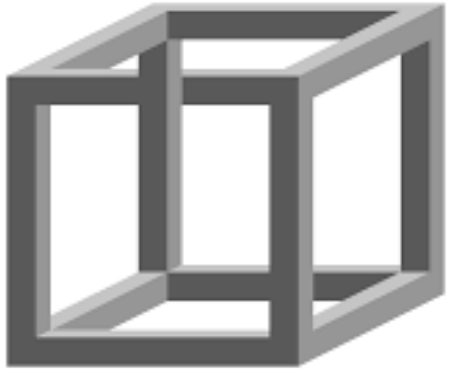
Rotations of such objects in 3D produce cross-terms in (xy, xz, yz). Parameters (A, B, C) determine object dimensions. Origin-centred.

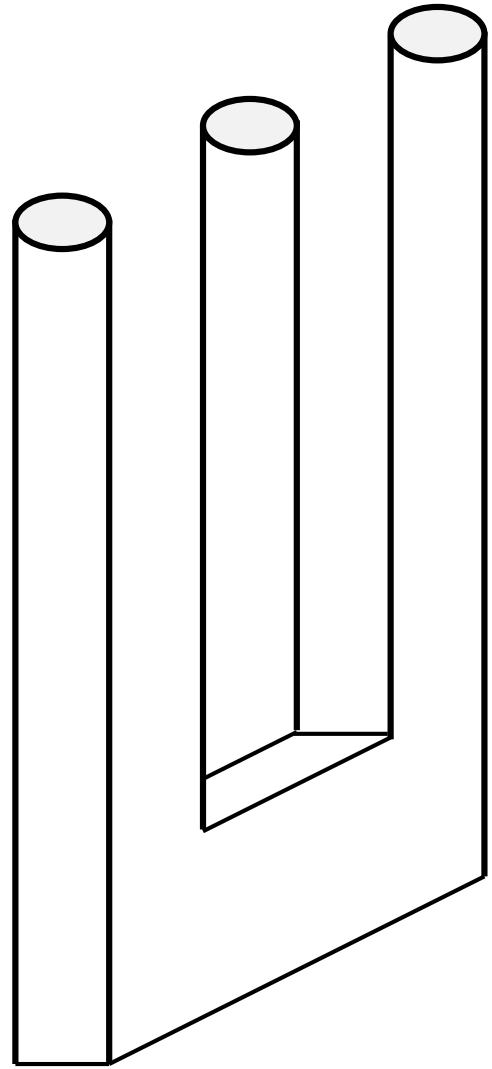
These simple, parametric models for solids, augmented by Boolean relations for conjoining them, allow the generation of object-centered, "volumetric" descriptions of many objects (instead of an image-based description) by just listing parameters $(\alpha, \beta, \gamma, A, B, C)$ and relations, rather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

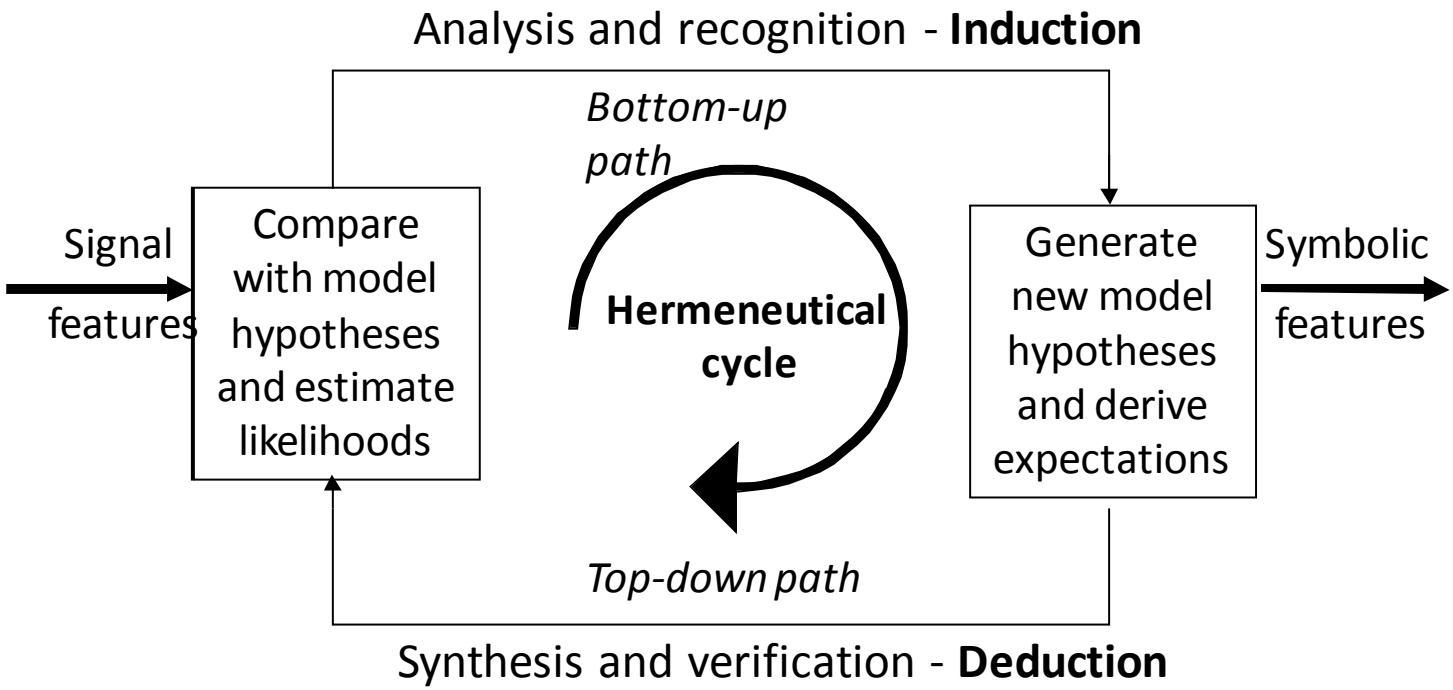






Vision as Perceptual Inference and Hypothesis Testing

- 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.

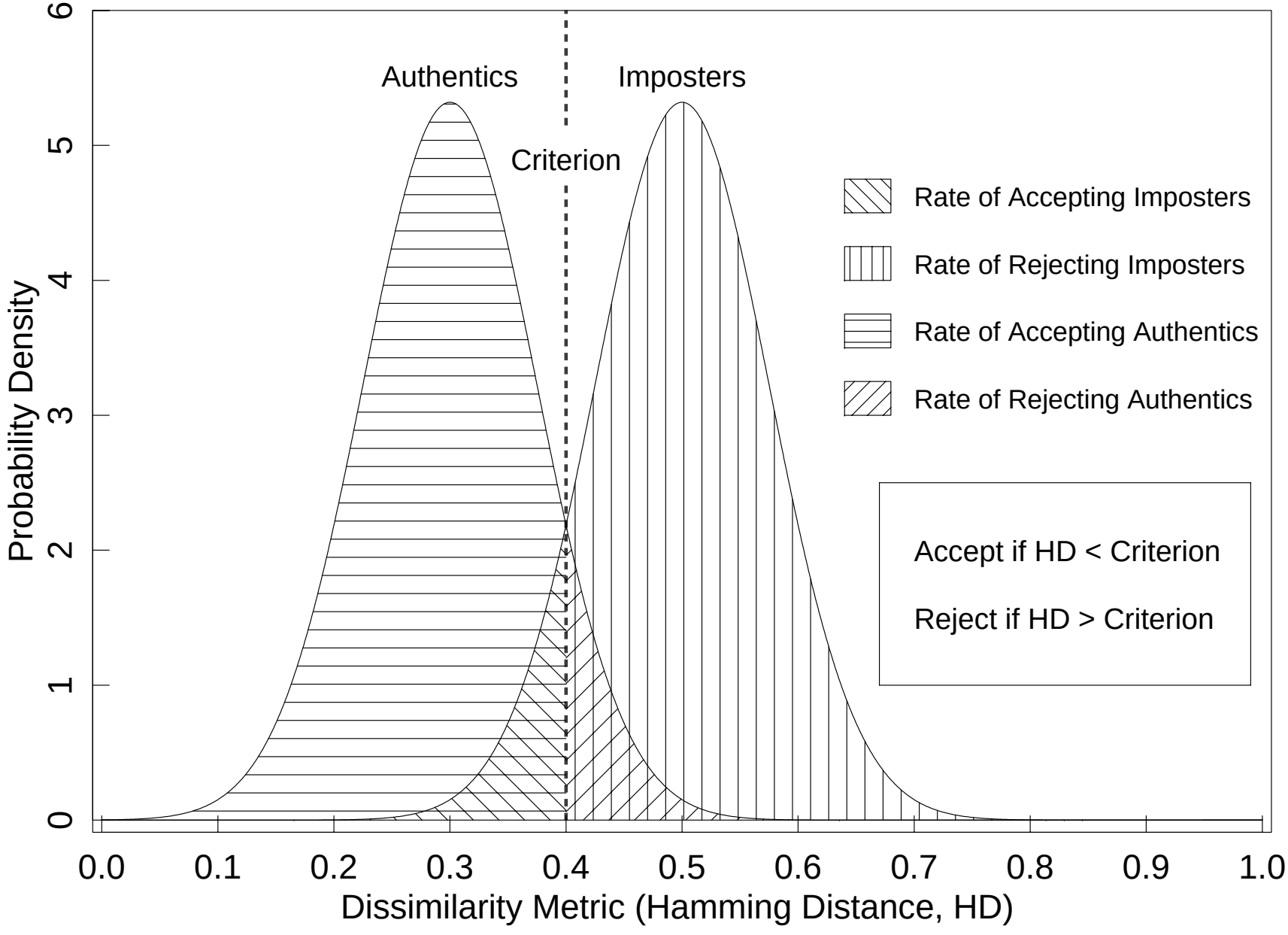
The task is to decide whether or not this feature vector is consistent with a particular class or object category. Thus the problem of classification amounts to a "same / different" decision about the presenting feature vector, compared with vectors characteristic of certain object classes.

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

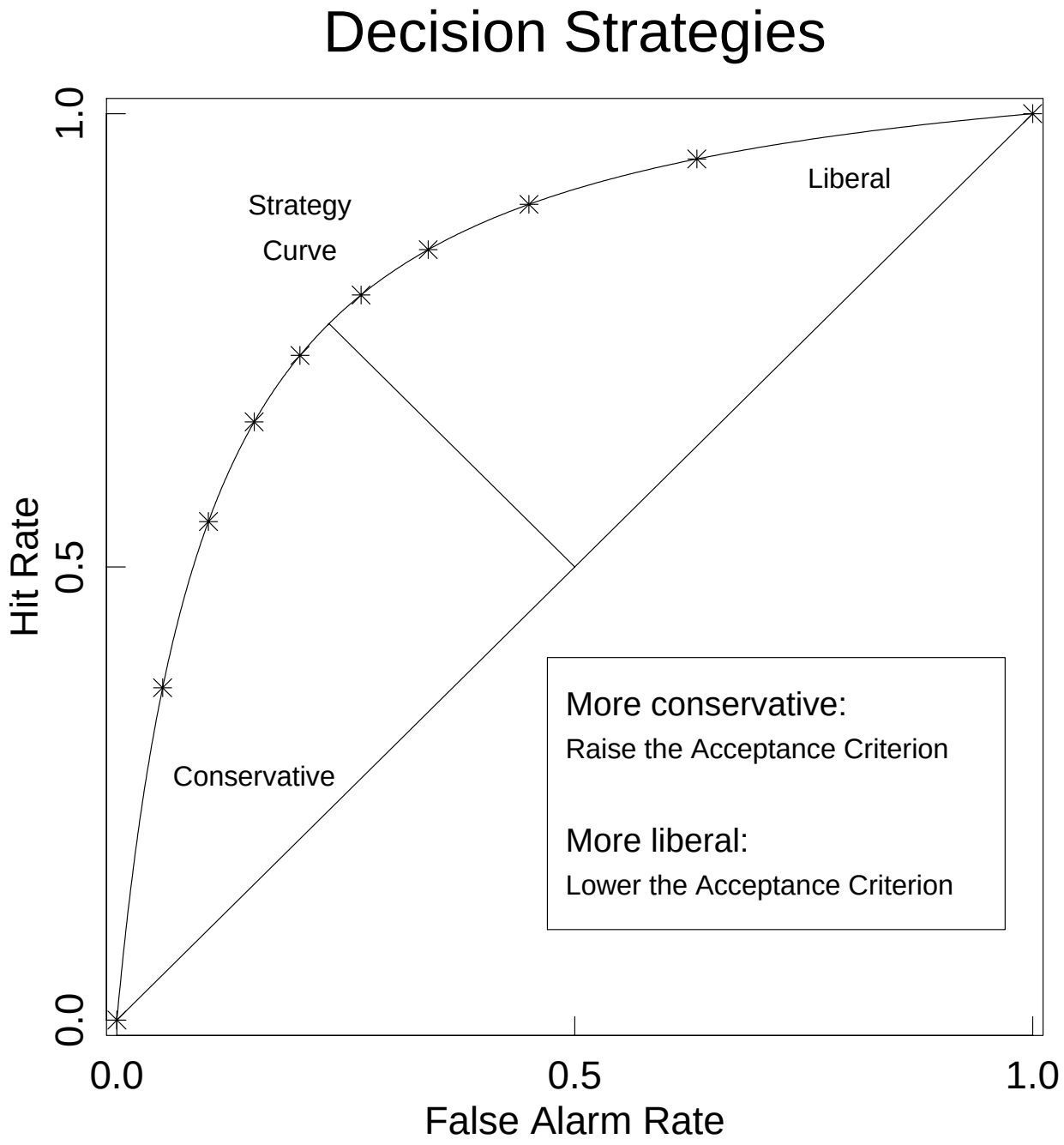
Statistical Decision Theory



(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):

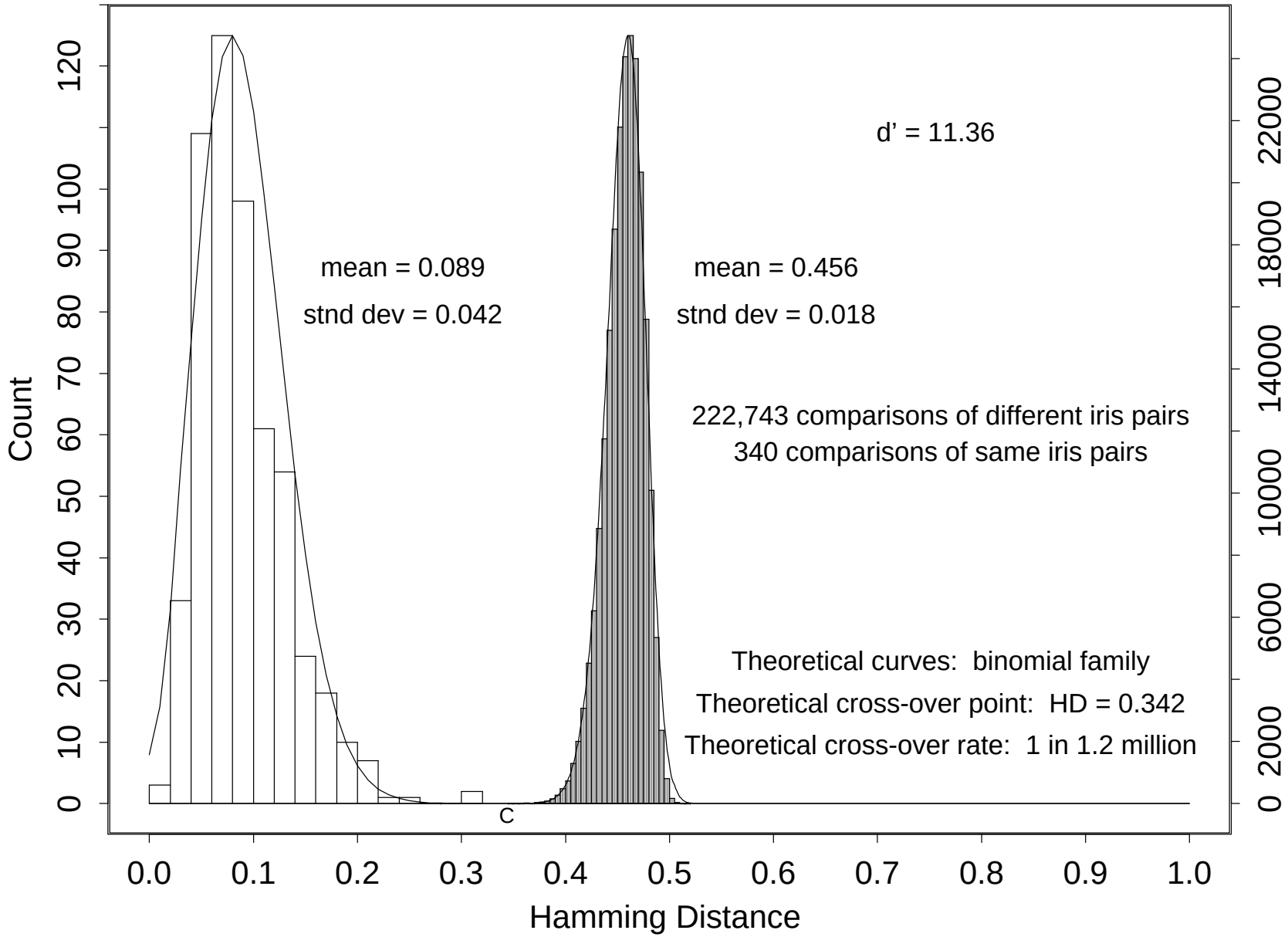
$$d' = \frac{|\mu_2 - \mu_1|}{\sqrt{\frac{1}{2}(\sigma_2^2 + \sigma_1^2)}}$$

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

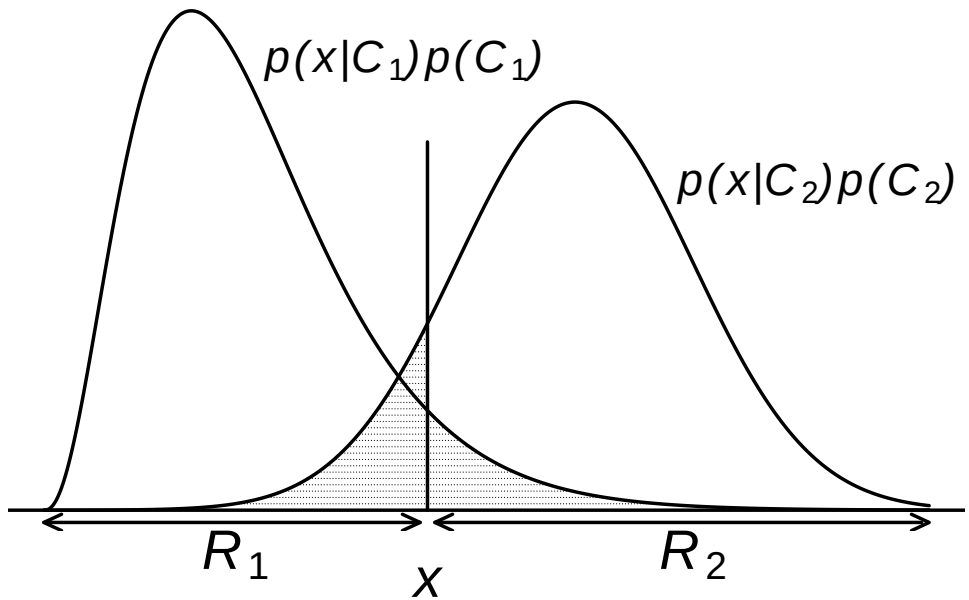


(Statistical Decision Theory, con't)

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 $d' \approx 11$.

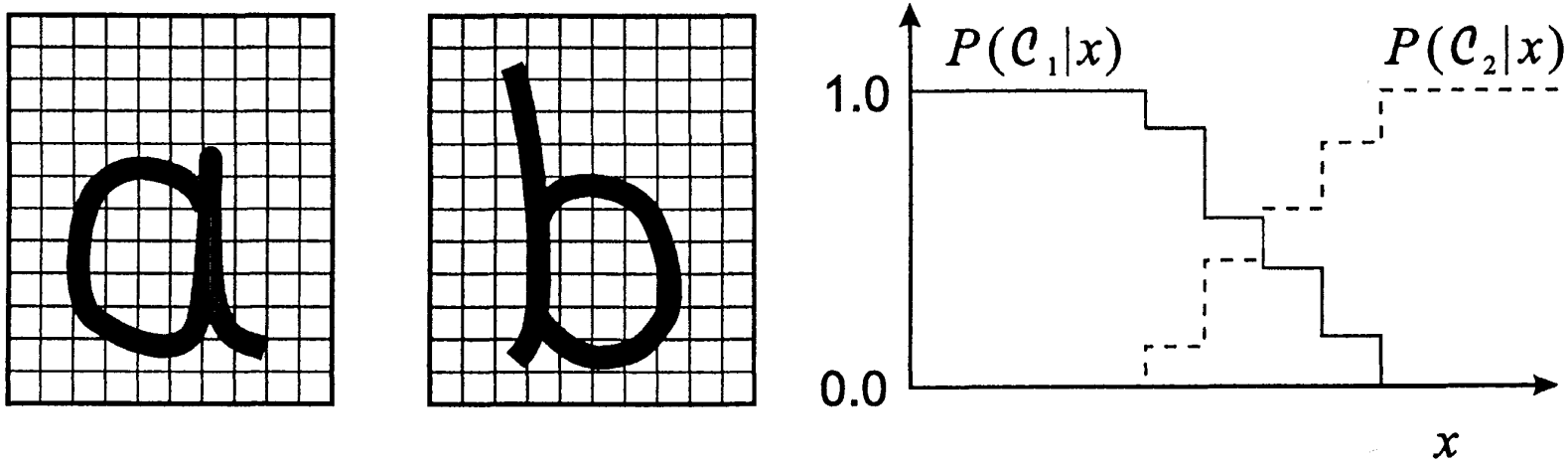
Because reliability of pattern recognition depends on the between-class variance being larger than the within-class variance, R. Fisher defined the "separation between two distributions" as the ratio of their between-class variance to their within-class variance. This definition is related to d'.

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:

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

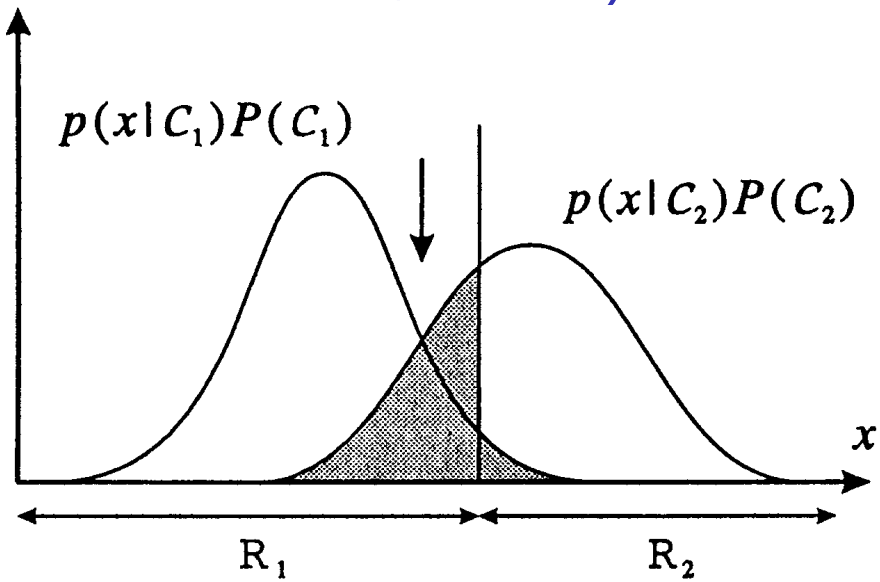
Since the denominator $P(x)$ in Bayes' Rule is independent of C_k, we can rewrite this minimum misclassification criterion simply as:

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:

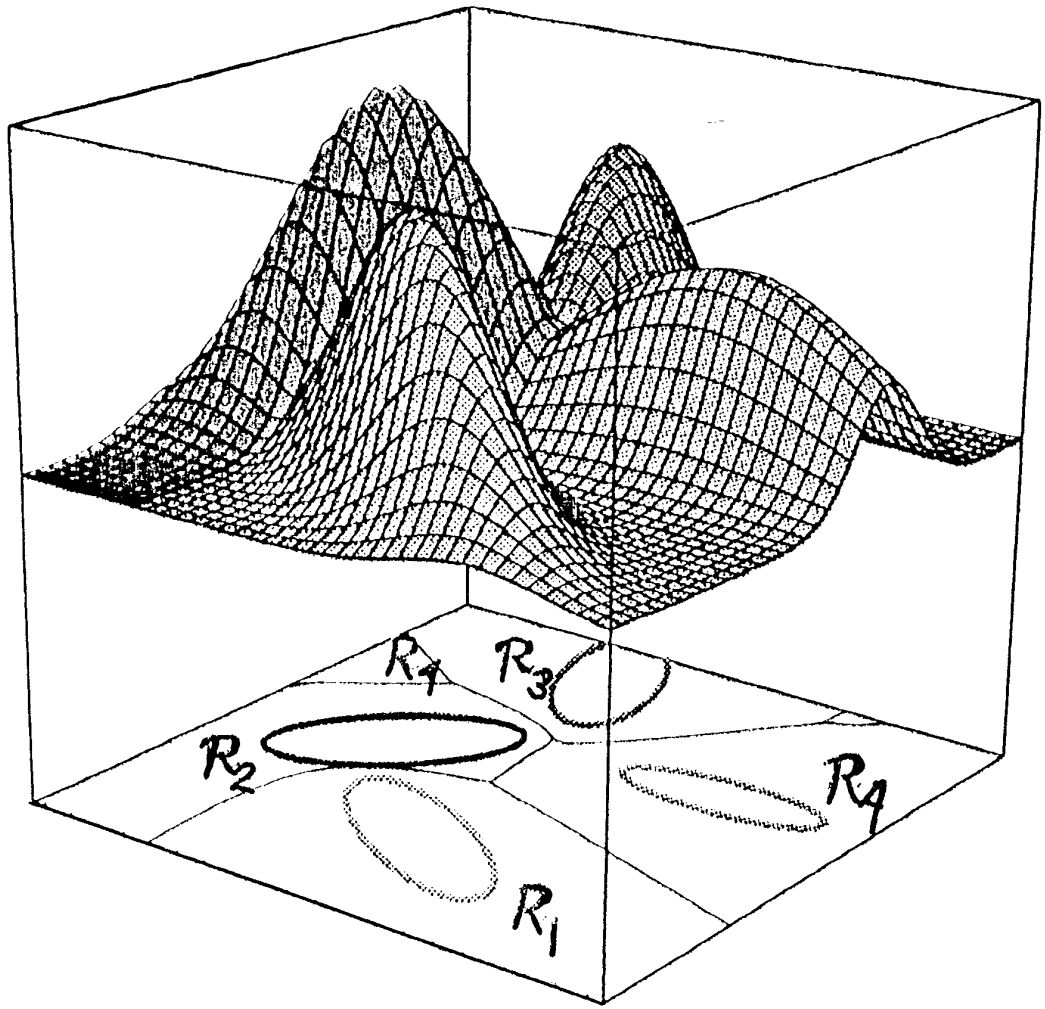
$$y_k(x) = P(C_k|x)$$

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.)



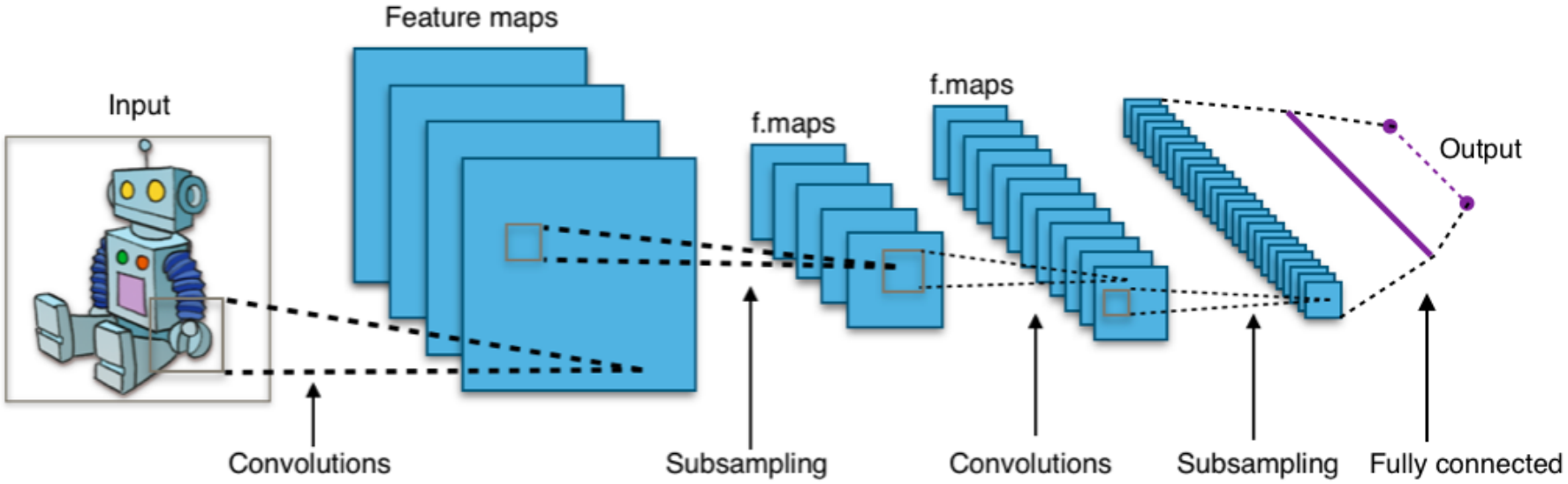
Discriminative versus Generative Methods in Vision

- 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)

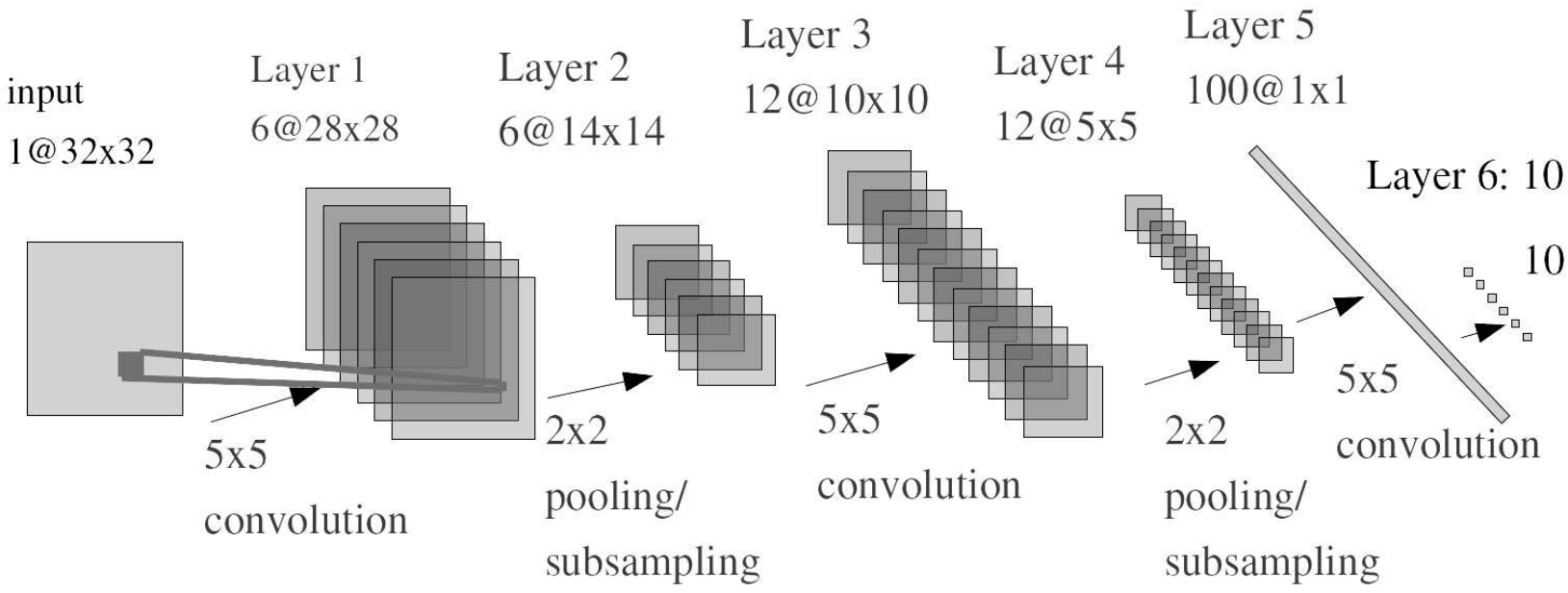


Example: Convolutional Neural Network for OCR (LeCun)

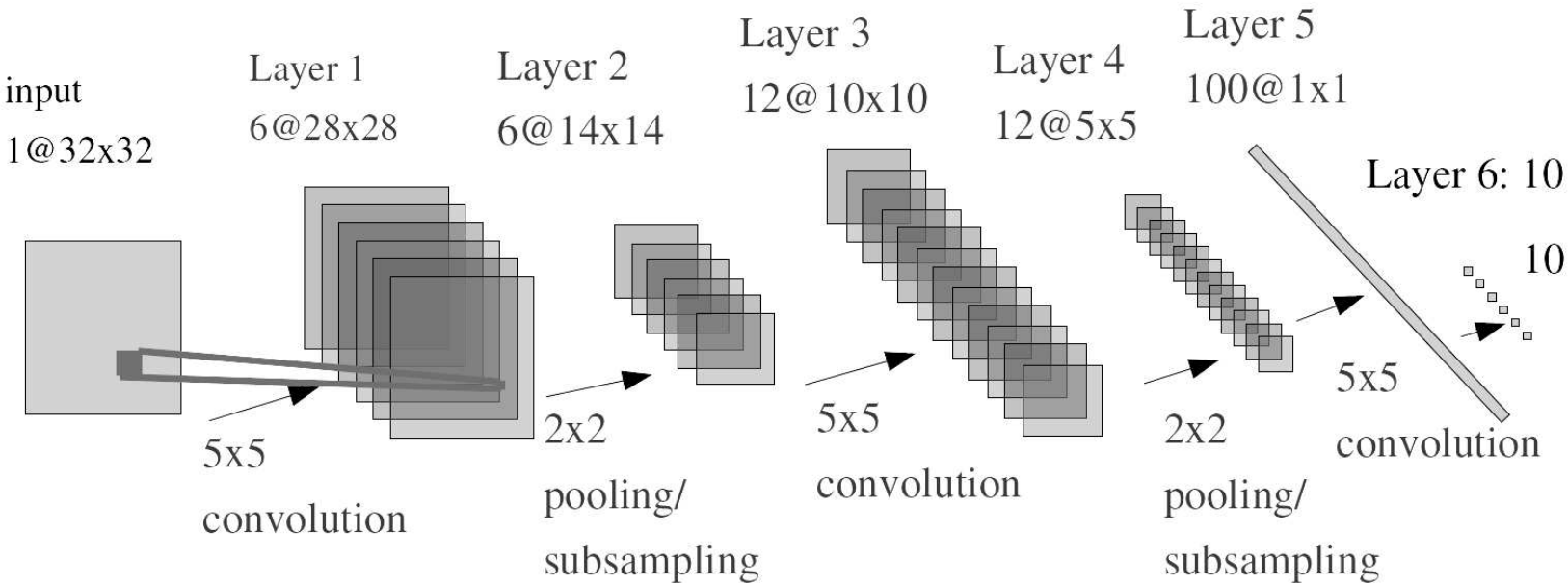
Optical Character Recognition systems have many applications:

- 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)



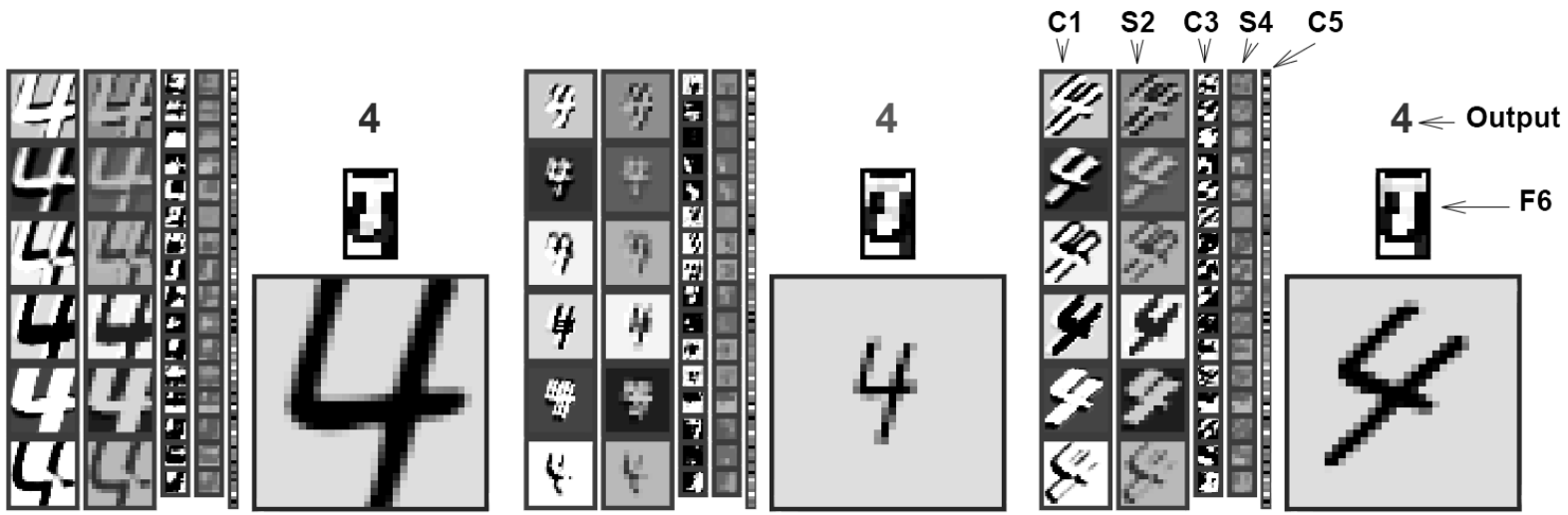
- Input is a 32×32 pixel image, containing some digit or character
- In the training phase, 100,000s of examples of each target are used
- Training is supervised back-propagation: target output is set to $+1$, all others to -1. Errors back-propagate to adaptable feature maps
- Neurons in a feature map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