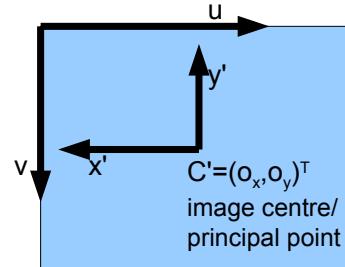


Content

- **Geometry of image formation**
 - perspective camera model
 - projective geometry
 - intrinsic and extrinsic parameters

Relating pixels to camera coordinates



To convert from camera reference frame [mm] to image reference frame [pixel], we need to take account of:

- Origin of image (in corner, not centre)
- Pixel size (s_x, s_y [mm/pixel])

$$u = \frac{-x'}{s_x} + o_x = \frac{-f' x}{s_x z} + o_x = \alpha \frac{x}{z} + o_x$$

$$v = \frac{-y'}{s_y} + o_y = \frac{-f' y}{s_y z} + o_y = \beta \frac{y}{z} + o_y$$

Where:

- u, v are integer image coordinates [pixel]

- α, β magnification factors in x' and y' directions ($\alpha = -f'/s_x$, $\beta = -f'/s_y$)

o_x, o_y, α, β are the 4 intrinsic camera parameters

α/β = aspect ratio of camera

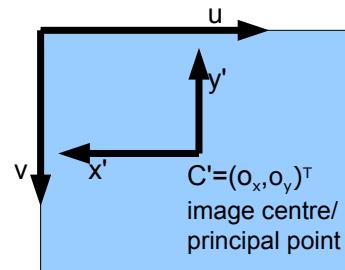
Computer Vision / Image Formation (Artificial and Biological)

Mapping from world to image coordinates

Where a point in the 3D world appears on an image depends not just on the camera model, but also on:

- **Intrinsic** parameters
 - Depend on properties of camera
 - focal length, sensor dimensions/resolution
- **Extrinsic** parameters
 - Depend on the camera location
 - translation and rotation of camera relative to world

Relating pixels to camera coordinates



We can re-write these equations in matrix form:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z} \begin{pmatrix} \alpha & 0 & o_x \\ 0 & \beta & o_y \\ 0 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\text{intrinsic camera parameters}} \underbrace{\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}}_{\text{projection operator}}$$

This is only an approximation, due to each individual camera having small manufacturing errors (i.e. the image plane may not be perfectly perpendicular to the optical axis, or may be rotated slightly about the optical axis ('skew'))

Computer Vision / Image Formation (Artificial and Biological)

Computer Vision / Image Formation (Artificial and Biological)

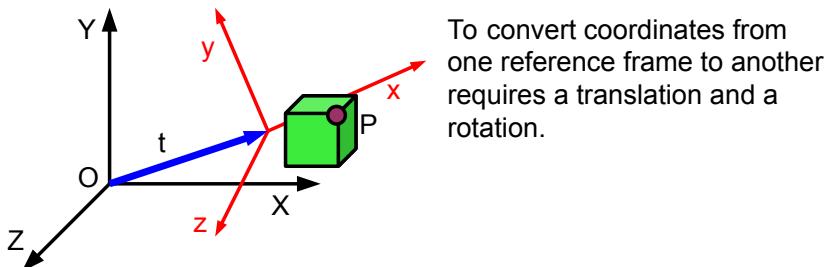
Relating pixels to world coordinates

So far it has been assumed that the location of the scene point $P=(x,y,z)^T$ is given in camera coordinates.

However, more generally scene points may be known relative to some external reference frame.

An external reference frame is especially useful when:

- the camera is moving
- multiple cameras are used (e.g. for stereopsis)



Computer Vision / Image Formation (Artificial and Biological)

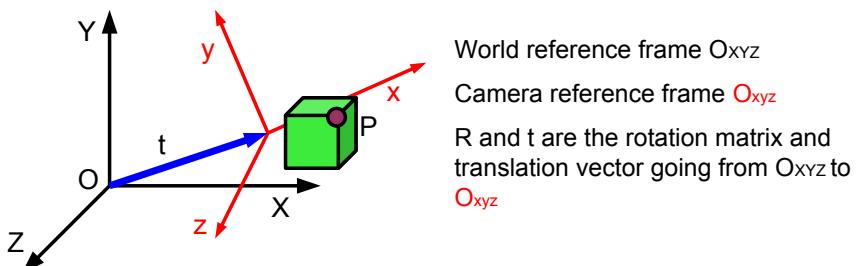
Relating pixels to world coordinates

Coordinates of point in frame O_{xyz}

Rotation matrix describing orientation of frame O_{xyz} with respect to frame O_{XYZ}

Coordinates of origin of frame O_{xyz} (in frame O_{XYZ})

Coordinates of point in frame O_{xyz}

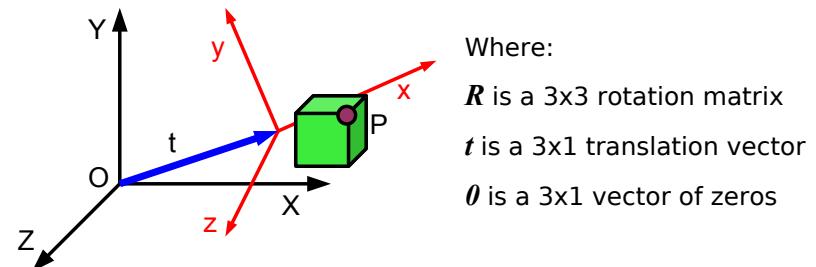


Computer Vision / Image Formation (Artificial and Biological)

Relating pixels to world coordinates

We can re-write these equations in matrix form:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

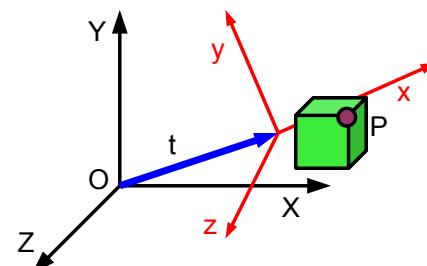


Computer Vision / Image Formation (Artificial and Biological)

Relating pixels to world coordinates

To go from the coordinates of a point in the world reference frame to the camera reference frame we require the inverse mapping:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} R^T & -R^T t \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$



Computer Vision / Image Formation (Artificial and Biological)

Relating pixels to world coordinates

The complete transformation is thus:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z} \begin{pmatrix} \alpha & 0 & o_x \\ 0 & \beta & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \underbrace{\begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix}}_{\text{extrinsic camera parameters}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

intrinsic camera parameters projection operator extrinsic camera parameters
2D → 2D 3D → 2D 3D → 3D

Maps points on the image plane into pixel image coordinates Projects points in the camera reference frame onto the image plane Transforms the world coordinates to the camera reference frame

Computer Vision / Image Formation (Artificial and Biological)

Mapping between 2D and 3D

Given a point in 3D space we can model where that point will appear in a 2D image.

» A well-posed, forward problem

However, given a point in a 2D image we cannot determine the corresponding point in 3D space (depth information has been lost).

» An ill-posed, inverse problem

To recover depth, need extra information – e.g.

- another image (the need for stereo vision), or
- prior knowledge about the structure of the scene.

Computer Vision / Image Formation (Artificial and Biological)

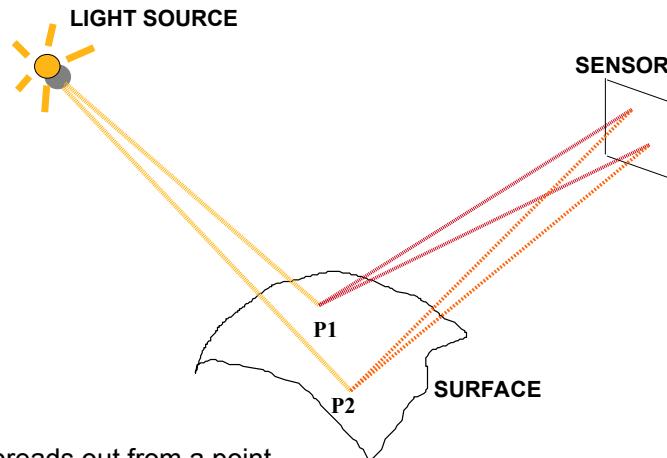
Content

• Physics of image formation

- light
- reflectance
- optics

Computer Vision / Image Formation (Artificial and Biological)

Focusing Light



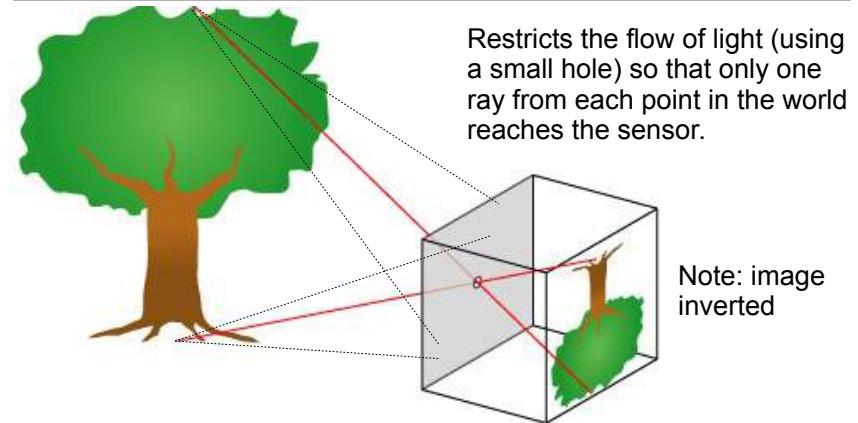
Light spreads out from a point.

Without some kind of optics each location on the sensor will register light coming from many different points in the world.

No image will be formed.

Computer Vision / Image Formation (Artificial and Biological)

Pinhole camera

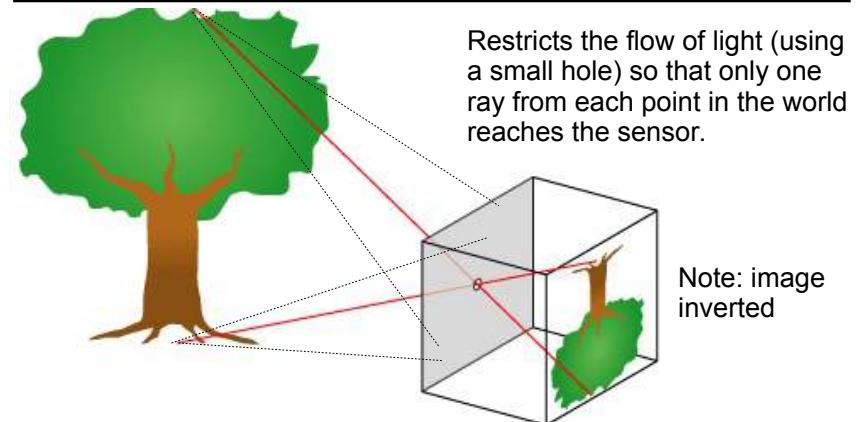


"Focus" means that all rays coming from a scene point converge into a single image point.

"Exposure" is the time needed to allow enough light through to form an image (the smaller the aperture, the longer the exposure time). The longer the exposure the more blurred an image is likely to be.

Computer Vision / Image Formation (Artificial and Biological)

Pinhole camera



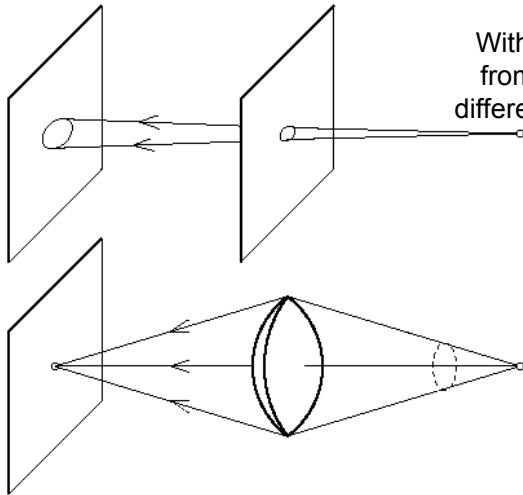
Small pinhole: sharp focus but dim image (long exposure time)

Large pinhole: brighter image (shorter exposure) but blurred

To produce an image that is both bright and in focus requires a lens

Computer Vision / Image Formation (Artificial and Biological)

Lensed camera



With a large pinhole, light rays from the same point project to different locations on the image. The image is blurred.

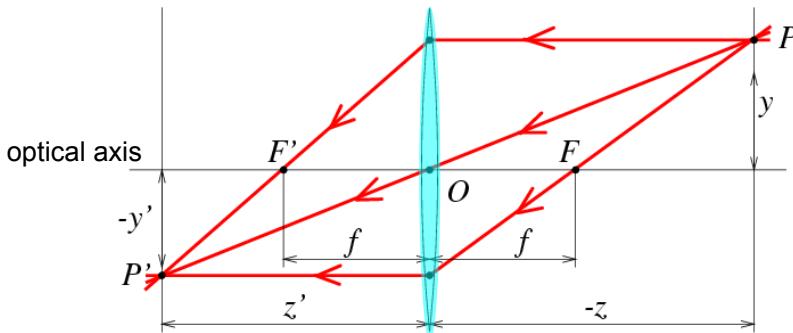
By focusing rays from the same point onto a single image location, a lens can keep the image sharp while gathering more light.

Cost: image focused for only a restricted range of object positions

Computer Vision / Image Formation (Artificial and Biological)

Thin lenses

A Lens works by refracting light.



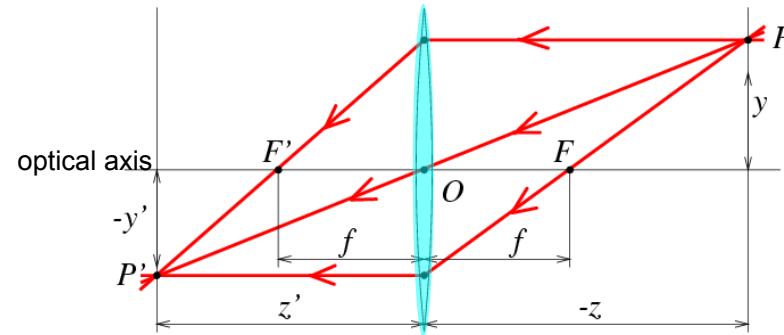
F' = focal point, f = focal length, O =optical centre

- Rays passing through O are not refracted
- Rays parallel to the optical axis are refracted to pass through F' .
- Rays passing through point F are refracted to be parallel to the optic axis.

f depends on the curvature of the lens and the material it is made from.

Computer Vision / Image Formation (Artificial and Biological)

Thin lenses



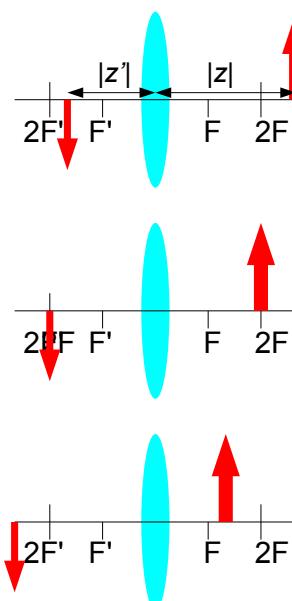
An object located at distance z from the lens has an image formed at depth at z' from the lens according to the “thin lens equation”:

$$\frac{1}{f} = \frac{1}{|z|} + \frac{1}{|z'|}$$

Hence, depth of the image depends on the depth of the object as well as the focal length of the lens.

Computer Vision / Image Formation (Artificial and Biological)

Depth of focus



$$\frac{1}{f} = \frac{1}{|z|} + \frac{1}{|z'|}$$

For a lens with a fixed focal length, the depth of the image plane required to bring an object into focus varies inversely with the depth of the object.

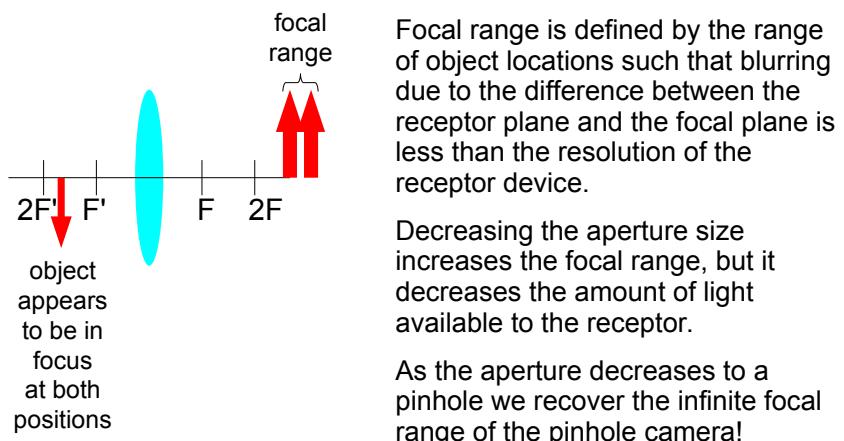
At the extremes: if the object is at infinity the image is formed at F' , if the object is at F (or closer) no image can be formed.

For a short focal length camera, almost all objects will be located more than 2 focal lengths from the lens (top figure)

Placing the receptor plane at a fixed depth in the range $z' \in [F', 2F']$ will provide an acceptable image for objects in a wide range of depths greater than $2F$.

Computer Vision / Image Formation (Artificial and Biological)

Focal range



Focal range is defined by the range of object locations such that blurring due to the difference between the receptor plane and the focal plane is less than the resolution of the receptor device.

Decreasing the aperture size increases the focal range, but it decreases the amount of light available to the receptor.

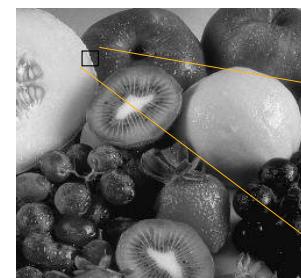
As the aperture decreases to a pinhole we recover the infinite focal range of the pinhole camera!

Content

- **Digital images**
 - digitisation
 - representation

Computer Vision / Image Formation (Artificial and Biological)

Digital image representation (greyscale)



A digital image is a 2D array (matrix) of numbers:

x =	58	59	60	61	62	63	64	
y =	41	7	10	10	15	50	70	80
42	1	0	67	123	25	30	130	
43	2	35	100	150	150	80	50	
44	8	15	70	100	10	20	20	
45	12	15	76	5	17	0	15	
46	5	15	50	120	110	130	110	
47	5	10	20	50	50	20	250	

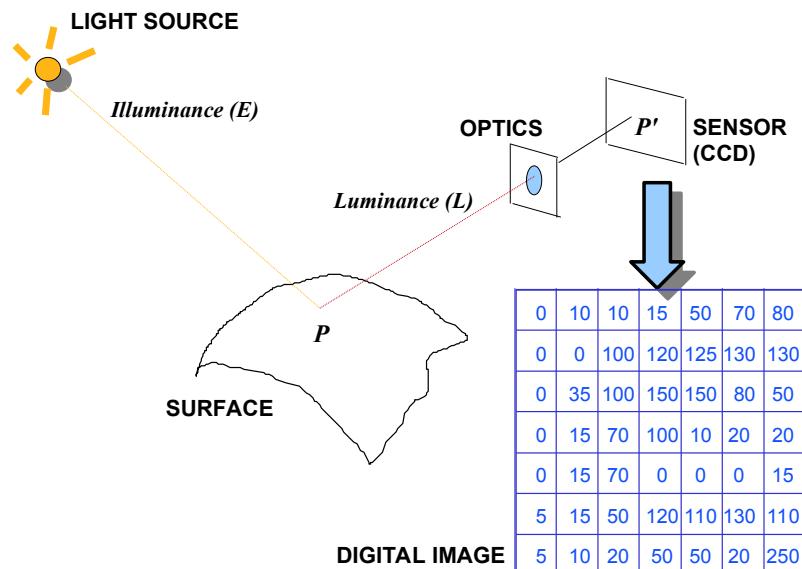
The scene, which is a continuous function, is sampled at discrete points (called *picture elements* or *pixels* for short).

Value of each pixel = measure of light intensity at that point.

Typically: Or:
0 = black 0 = black
255 = white 1 = white
integers floats

Computer Vision / Image Formation (Artificial and Biological)

Digital image formation



Computer Vision / Image Formation (Artificial and Biological)

Image axes and notation

A digital image is usually denoted as I .

- I is a matrix of pixel values
- Origin is top left corner

A point on the image, $p = (x,y)^T$

- p is a vector

A pixel value is $I(p)$ or $I(x,y)$

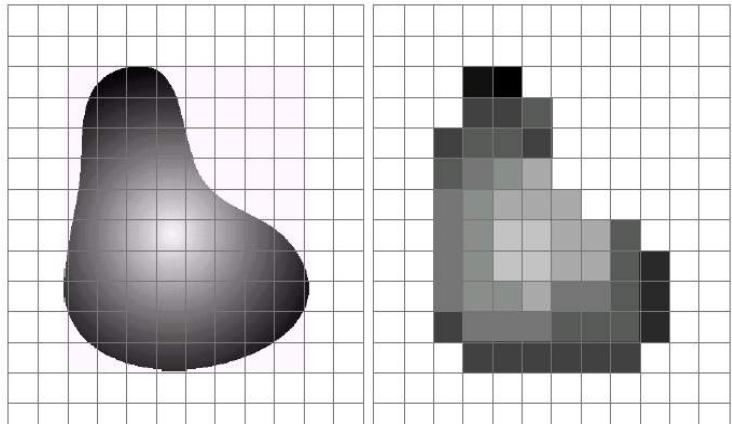
Note: in mathematics (and Matlab)

$I(r,c)$ would refer to the r^{th} row and the c^{th} column of I , so I is the transpose of a matrix in standard format.

x =	58	59	60	61	62	63	64	
y =	41	7	10	10	15	50	70	80
42	1	0	67	123	25	30	130	
43	2	35	100	150	150	80	50	
44	8	15	70	100	10	20	20	
45	12	15	76	5	17	0	15	
46	5	15	50	120	110	130	110	
47	5	10	20	50	50	20	250	

Computer Vision / Image Formation (Artificial and Biological)

Pixelisation and Quantization

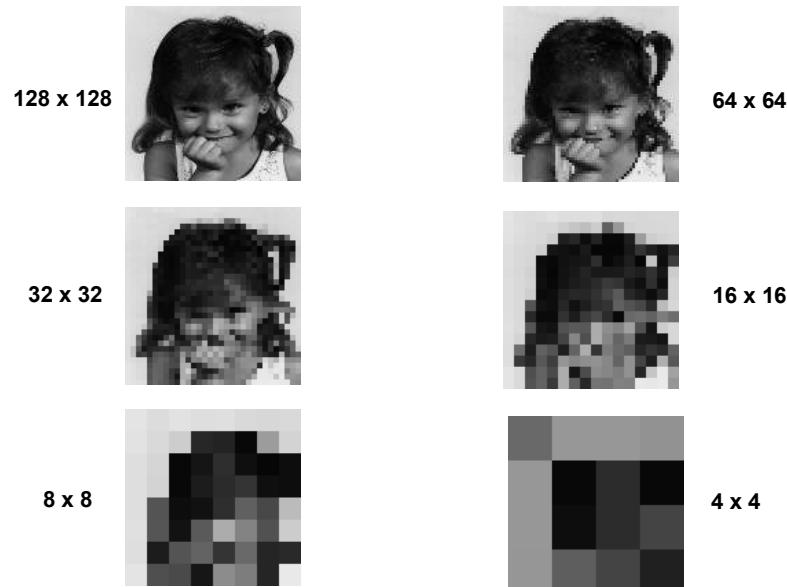


Pixelisation: intensity values averaged at each sampling point (i.e. within each grid location in the sensor array).

Quantization: intensity values represented using a finite number of discrete values (e.g. rounded to the nearest integer value).

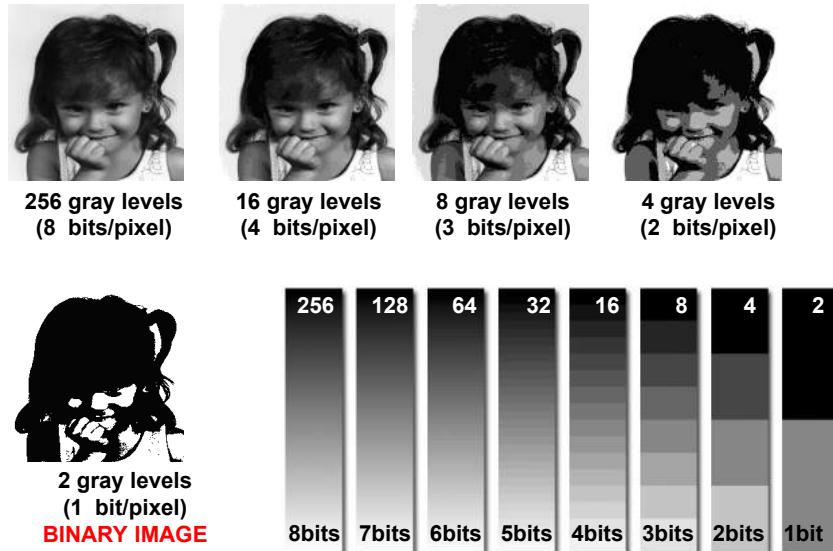
Computer Vision / Image Formation (Artificial and Biological)

Effects of pixelization



Computer Vision / Image Formation (Artificial and Biological)

Effects of quantization



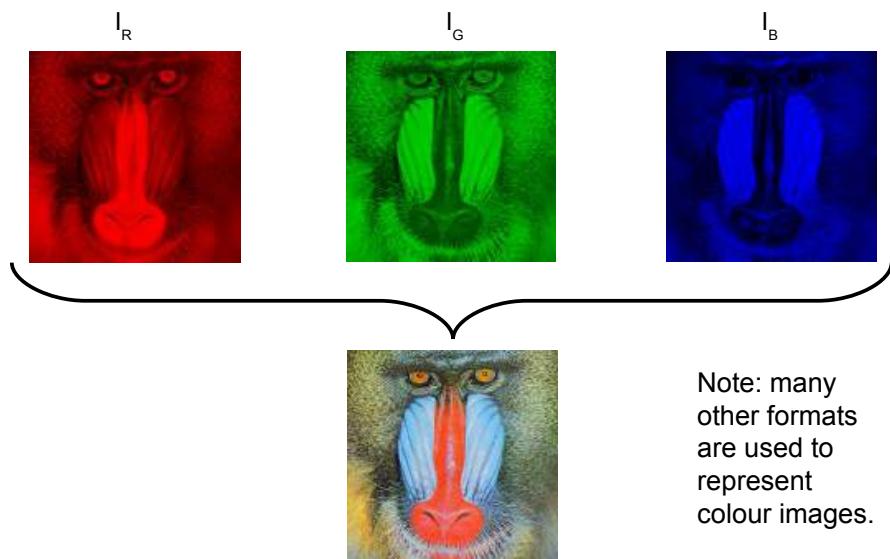
Computer Vision / Image Formation (Artificial and Biological)

Image formats

- Binary or Monochrome:
 - 1 binary value per pixel, 1-bit \in 2 discrete levels.
 - $I(p) = 0$ or 1 .
- Greyscale or Intensity:
 - 1 real value per pixel, typically 8-bit \in 256 discrete levels.
 - $I(p) \in [0,1]$
- Colour:
 - 3 real values per pixel, i.e. 3 colour channels, e.g. RGB
 - $I_R(p) \in [0,1]$, $I_G(p) \in [0,1]$, $I_B(p) \in [0,1]$
 - Each colour channel is a greyscale image representing the intensity of light at a particular wavelength ($R=645.2\text{nm}$, $G=526.3\text{nm}$, $B=444.4\text{nm}$)
 - 24-bit 'True Color' \in 8-bits for each colour.

Computer Vision / Image Formation (Artificial and Biological)

Colour image representation (RGB)



Note: many other formats are used to represent colour images.

Computer Vision / Image Formation (Artificial and Biological)

Switching between formats

RGB to Greyscale

Take weighted average over three colour channels:

$$I_{grey} = \frac{rI_R + gI_G + bI_B}{r + g + b}$$

where r, g, b are three weighting factors (if r=g=b, this is simply the mean).

Greyscale to binary

Apply a threshold t:

$$I_{bin}(p) = \begin{cases} 1 & \text{if } I_{grey}(p) \geq t \\ 0 & \text{otherwise} \end{cases}$$

Computer Vision / Image Formation (Artificial and Biological)

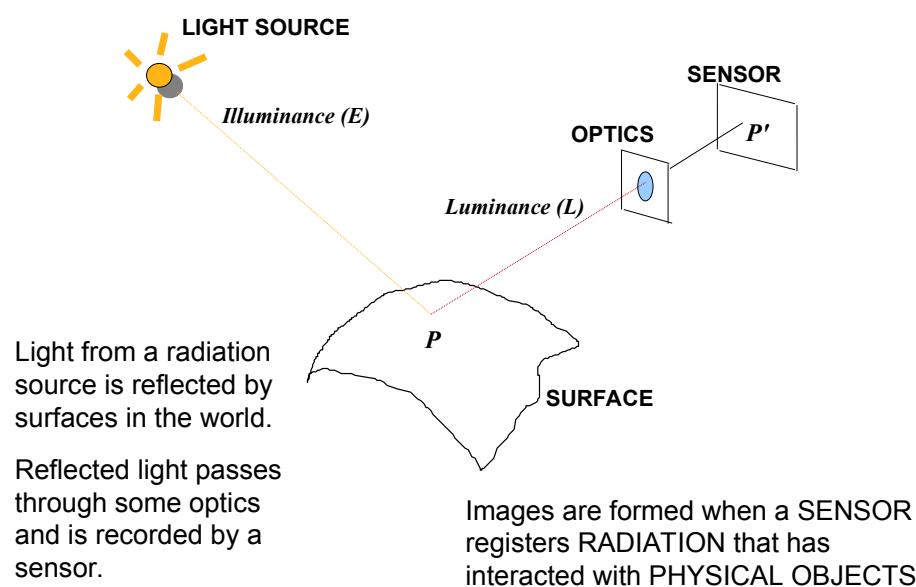
Content

• Physics of image formation

- light
- reflectance
- optics

Computer Vision / Image Formation (Artificial and Biological)

Overview of image formation



Ingredients of image formation

The image that is formed is affected by two sets of parameters:

Radiometric parameters

Determine the intensity/colour of a given location in the image

- illumination (type, number, location, intensity, colour-spectrum)
- surface reflectance properties (material, orientation)
- sensor properties (sensitivity to different electromagnetic frequencies)

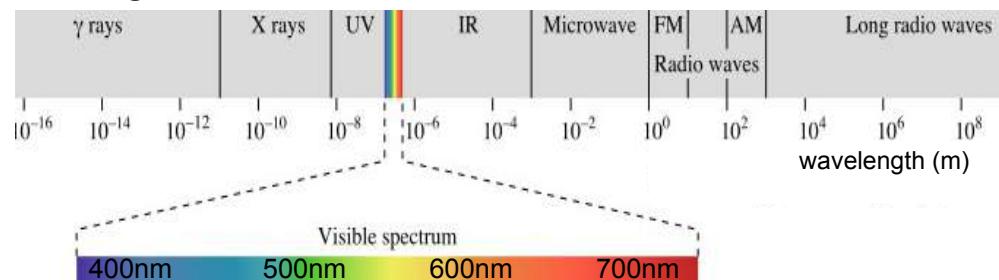
Geometric parameters

Determine where on the image a scene point appears

- camera position and orientation in space
- camera optics (e.g. focal length)
- projection geometry (mapping from 3D to 2D)

Computer Vision / Image Formation (Artificial and Biological)

Light and Colour



- At the earth's surface the intensity of the electromagnetic radiation emanating from the sun has a peak within the 400-700nm range.
- The human eye has evolved a specific sensitivity to this part of the electromagnetic spectrum.
- Hence, visible light is that part of the electromagnetic spectrum with a wavelength (λ) between 400 and 700nm.
- Cameras and Computer Vision systems also concentrate on this part of the spectrum (but not exclusively, e.g. infra-red cameras for detecting body heat, X-rays for imaging inside the body).

Computer Vision / Image Formation (Artificial and Biological)

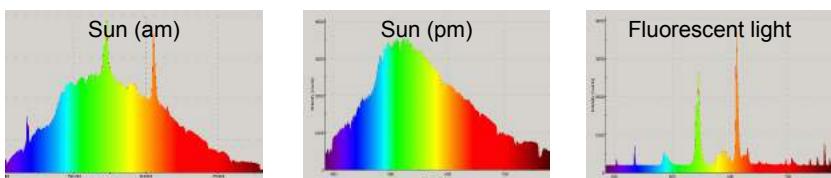
Colour perception

- The radiation that drives the human construct of colour, is fundamentally colour less.
- The sensation of colour is determined by the human visual system, based on the product of light and reflectance.
- However, it is a convenient short-hand to refer to electromagnetic radiation as having colour, e.g. to say “red-light” or “blue-light”.

Computer Vision / Image Formation (Artificial and Biological)

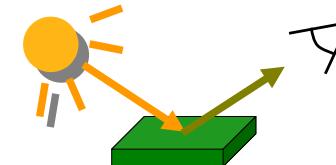
Illuminance

Light is produced in different amounts at different wavelengths by each light source.



Computer Vision / Image Formation (Artificial and Biological)

Luminance



Light is differentially reflected at each wavelength, which gives objects their natural colours.

albedo = fraction of light reflected at a particular wavelength

An object looks green because it absorbs red and blue light leaving more green in the reflected light.

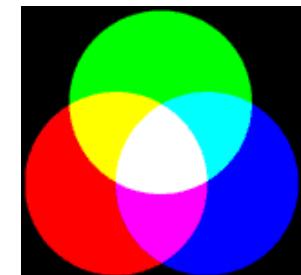
Computer Vision / Image Formation (Artificial and Biological)

Colour mixing

Mixing light: **additive**

e.g. a green light plus a blue light plus a red light gives light containing a broad spectrum of light, i.e. white.

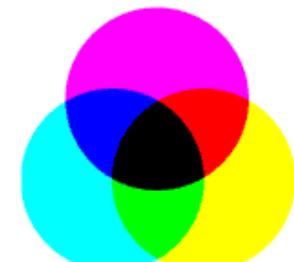
The illumination from different light sources adds.



Mixing pigments: **subtractive**

e.g. a green pigment plus a blue pigment plus a red pigment gives a pigment that **absorbs** light over a broad spectrum, leaving black.

The reflection from different surfaces subtracts.

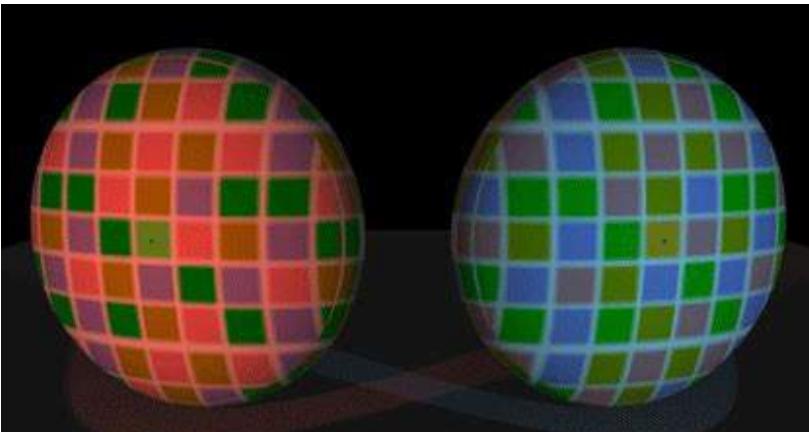


Computer Vision / Image Formation (Artificial and Biological)

Measuring surface properties

The biological vision system perceives the colour of surfaces in the world, not the colour of the light entering the eyes.

e.g.:



Looks green-blue

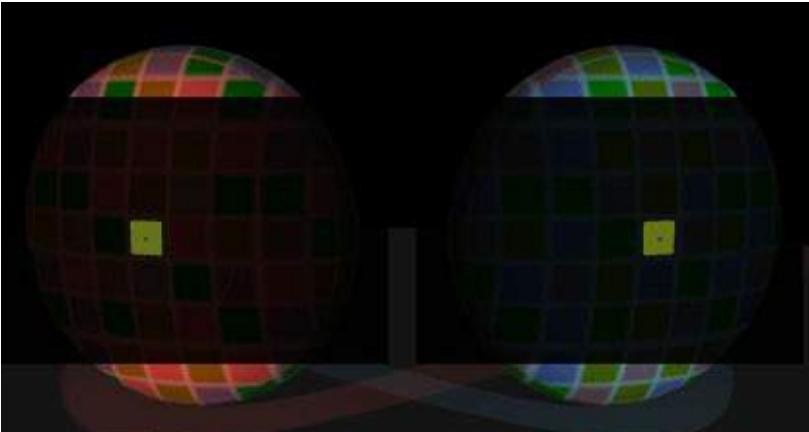
Looks yellow-orange

Computer Vision / Image Formation (Artificial and Biological)

Measuring surface properties

The biological vision system perceives the colour of surfaces in the world, not the colour of the light entering the eyes.

e.g.:



Light entering eyes is green-yellow

Computer Vision / Image Formation (Artificial and Biological)

Measuring surface properties

The biological vision system perceives the colour of surfaces in the world, not the colour of the light entering the eyes.

This is an ill-posed problem: we record L but need to recover R and don't know E.

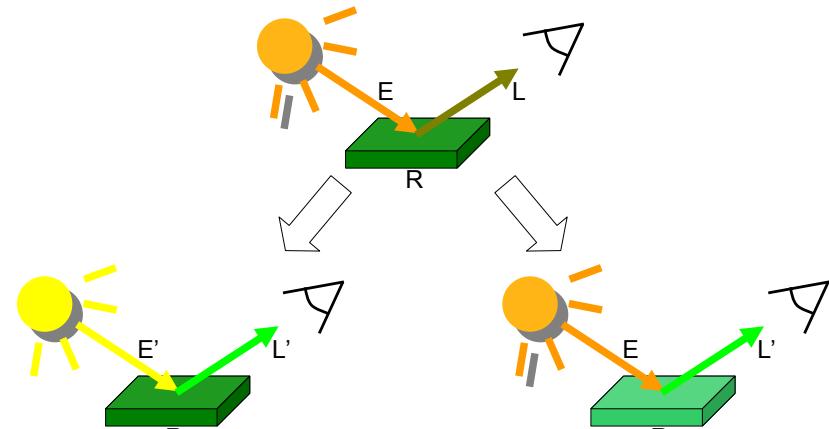
$$L(x,y,\lambda) = f_n(E(x,y,\lambda), \underbrace{R(x,y,\lambda)}_{\text{Intensity at a particular location and wavelength}})$$

Intensity at a particular location and wavelength

Luminance (L) amount of light striking the sensor, depends on **Illuminance** (E) amount of light striking the surface, as well as **Reflectance** (R) which depends on material properties

Computer Vision / Image Formation (Artificial and Biological)

Measuring surface properties



If colour/intensity of light source changes, colour/intensity of reflected light also changes (i.e. L varies with E)

If colour/reflectance of surface changes, colour/intensity of reflected light also changes (i.e. L varies with R)

Computer Vision / Image Formation (Artificial and Biological)

Colour constancy: artificial

To recover the surface colour of a particular location, $R(x,y,\lambda)$, we need to know the colour of the illumination at that point, $E(x,y,\lambda)$.

Many ways of approximating E have been suggested:

- Average reflectance across scene is known (often fails)
- Fixing brightest image patch to be white
- Gamut (collection of all colours) falls within known range
- Known reference colour (colour chart, skin colour...)
- Specular reflections have the colour of the illumination

None of these work particularly well.

Computer Vision / Image Formation (Artificial and Biological)

Colour constancy: biological

However, the human visual system does seem able to recover surface colour, since despite large changes in illumination (and consequently the intensity spectrum that enters our eyes), we usually experience the colour of an object as being constant.



We are not normally aware of this variation because colour constancy mechanisms discount the effects of illumination, and infer the colour of the objects.

Computer Vision / Image Formation (Artificial and Biological)

Content

- **Geometry of image formation**
 - perspective camera model
 - projective geometry
 - intrinsic and extrinsic parameters

Computer Vision / Image Formation (Artificial and Biological)

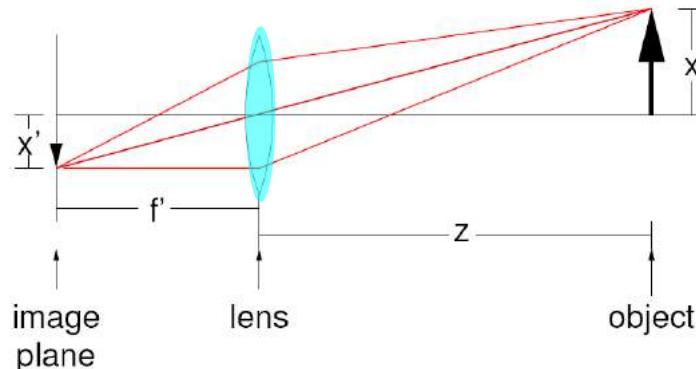
Geometric camera models

Given the coordinates of a point in the scene, what are the coordinates of this point in the image?

i.e. given $P = (x, y, z)$ how do we calculate $P' = (x', y', z')$?

To answer, we need a mathematical model of the geometric projection implemented by the camera, often called simply a camera model.

Perspective (or pinhole) camera model



A lens follows the pinhole model for objects that are in focus.

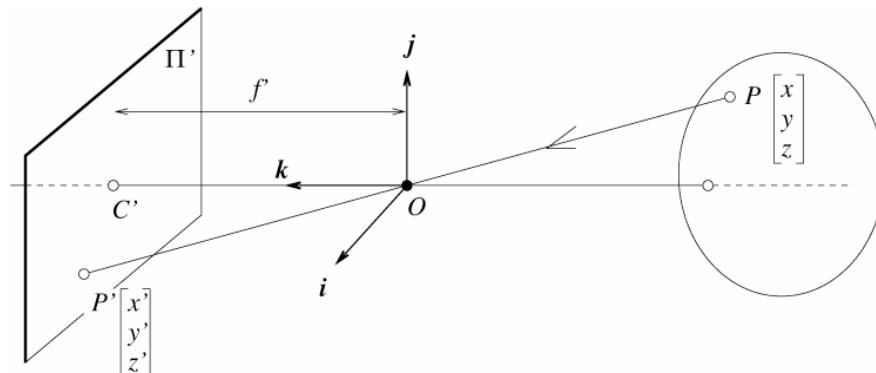
The pinhole camera is therefore an acceptable mathematical approximation (i.e. a model) of image formation in a real camera.

For a pinhole camera everything is in focus regardless of image plane depth

Pinhole model arbitrarily assumes that image plane is at distance f'

Computer Vision / Image Formation (Artificial and Biological)

Perspective camera model



P : a scene point with coordinates (x, y, z) , P' : its image with coordinates (x', y', z')

O : origin (pin hole / centre of lens)

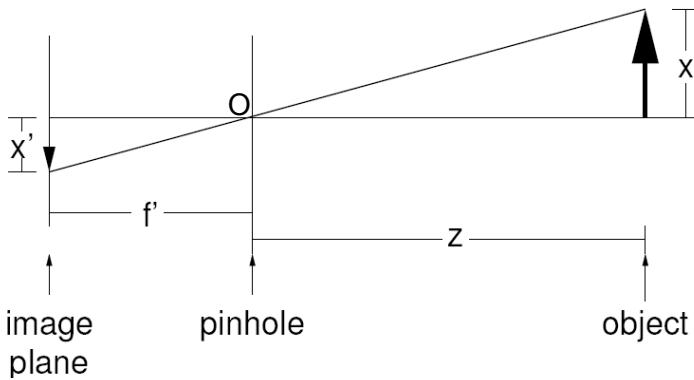
Image plane Π' is located at a distance f from the pinhole along the vector k
optical axis: line perpendicular to image plane and passing through O

C' : image centre (intersection of optical axis and image plane)

Computer Vision / Image Formation (Artificial and Biological)

Computer Vision / Image Formation (Artificial and Biological)

Equation of projection (2D)



From similar triangles:

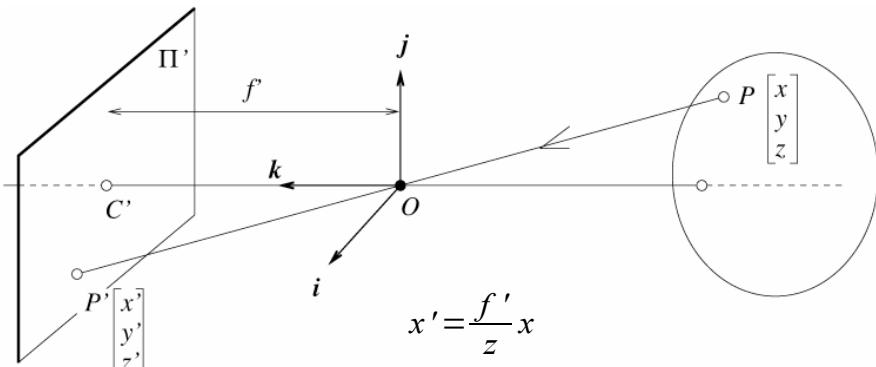
$$x'/x = f'/z$$

Hence:

$$x' = (f'/z) x$$

Computer Vision / Image Formation (Artificial and Biological)

Equation of projection (3D)



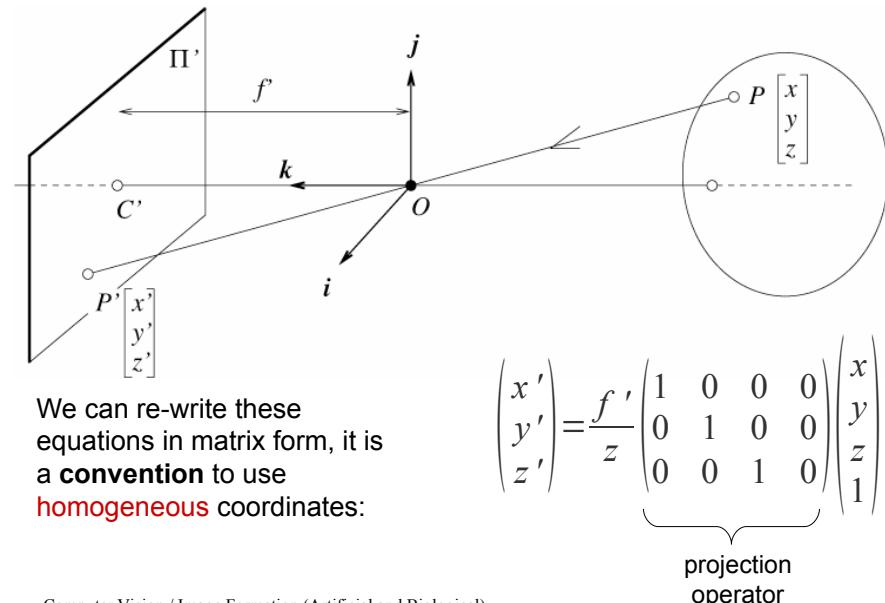
$$\text{Similarly, for } y \text{ coordinates: } y' = \frac{f'}{z} y$$

$$\text{Since } P' \text{ lies on image plane: } z' = f'$$

All coordinates are relative to camera reference frame [mm] – i.e.
axes rigidly attached to camera with origin at pinhole

Computer Vision / Image Formation (Artificial and Biological)

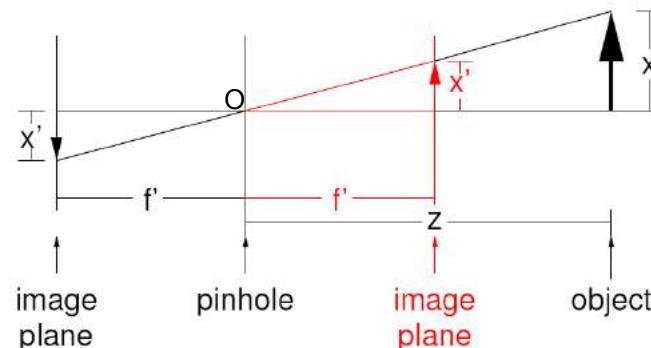
Equation of projection (3D)



We can re-write these equations in matrix form, it is a **convention** to use **homogeneous coordinates**:

Computer Vision / Image Formation (Artificial and Biological)

Virtual image



A pinhole camera creates inverted images.

It is traditional to draw the image plane in front of the pinhole at the same distance from it as the actual image plane.

Resulting “virtual” image is identical to the real image except it is the right way up.

This does **not** change the mathematics of the perspective camera model.

Computer Vision / Image Formation (Artificial and Biological)

Projective geometry

Euclidean geometry describes objects “as they are”.

It describes transformations within a 3D world (i.e. translations and rotations)

These mappings do not change the shape of an object (i.e. lengths, angles, and parallelism are preserved).

Projective geometry describes objects “as they appear”.

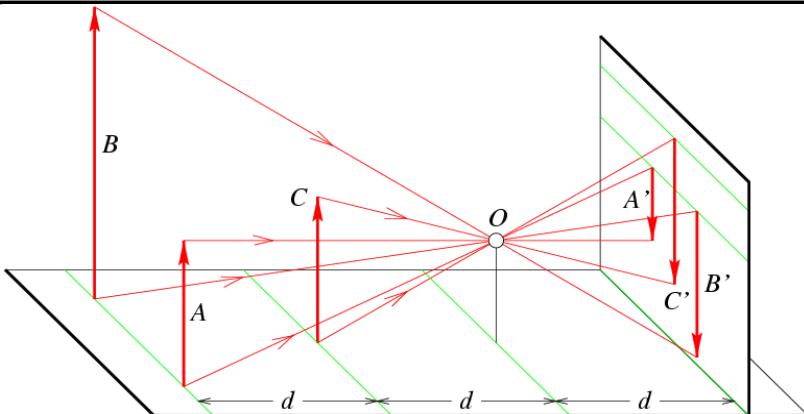
It describes the transformation from the 3D world to a 2D image (i.e. scaling and shear in addition to translations and rotations)

This mapping does distort the shape of an object (i.e. lengths, angles, and parallelism are not preserved).

Some properties of (i.e. distortions caused by) projective geometry are described on the following slides...

Computer Vision / Image Formation (Artificial and Biological)

Distant objects appear smaller



The apparent size of an object depends on its distance
In world:

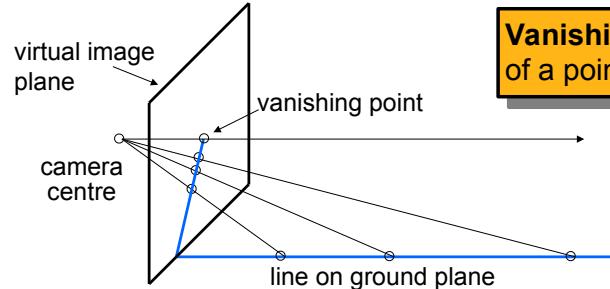
A and C have equal length, B twice this length

In image:

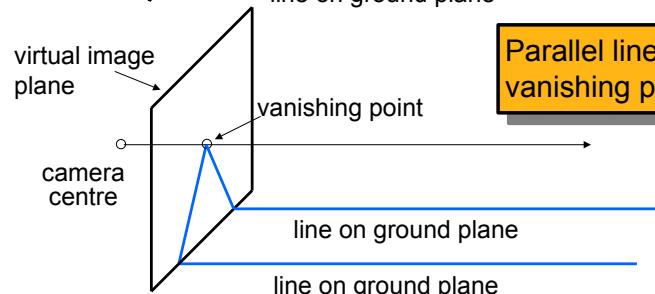
B' and C' have equal length, A' half this length

Computer Vision / Image Formation (Artificial and Biological)

Vanishing points



Vanishing point = projection of a point at infinity



Parallel lines have the same vanishing point

Computer Vision / Image Formation (Artificial and Biological)

Example of projective distortion

Reality is distorted by projection:

In world:

- rail tracks parallel
- ties equal in length
- ties perpendicular to tracks
- ties evenly spaced



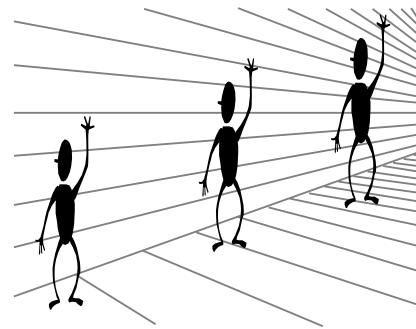
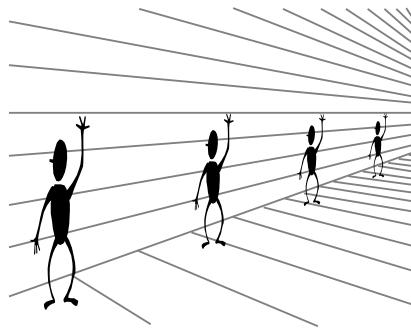
In image:

- rail tracks converge at a vanishing point
- ties get shorter with distance
- ties not at right angles to track
- ties get closer together at longer distances.

Computer Vision / Image Formation (Artificial and Biological)

Vanishing points provide cues to size

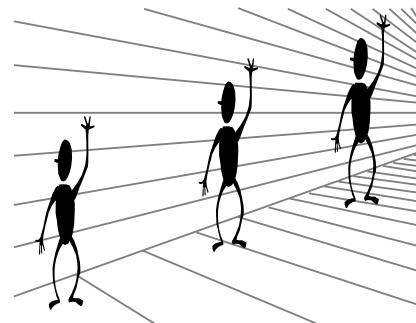
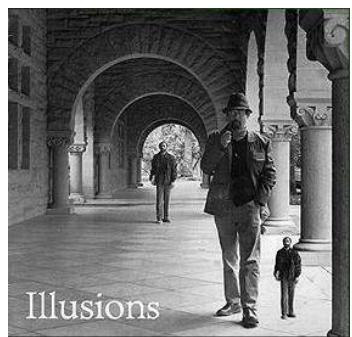
Our visual system is good at detecting vanishing points and using this to extract information about depth and object size.
Can be used in computer vision too.



Computer Vision / Image Formation (Artificial and Biological)

Vanishing points provide cues to size

Our visual system is good at detecting vanishing points and using this to extract information about depth and object size.
Can be used in computer vision too.



Computer Vision / Image Formation (Artificial and Biological)

Content

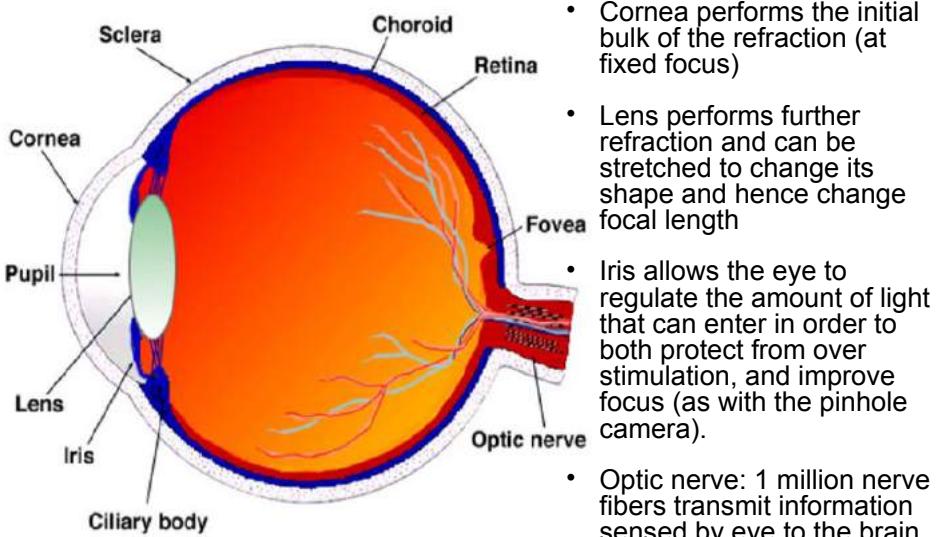
The Eye

- photoreceptors
- sampling
- ganglion cells
- image processing



Computer Vision / Image Formation (Artificial and Biological)

The Eye



- Cornea performs the initial bulk of the refraction (at fixed focus)
- Lens performs further refraction and can be stretched to change its shape and hence change focal length
- Iris allows the eye to regulate the amount of light that can enter in order to both protect from over stimulation, and improve focus (as with the pinhole camera).
- Optic nerve: 1 million nerve fibers transmit information sensed by eye to the brain.

Computer Vision / Image Formation (Artificial and Biological)

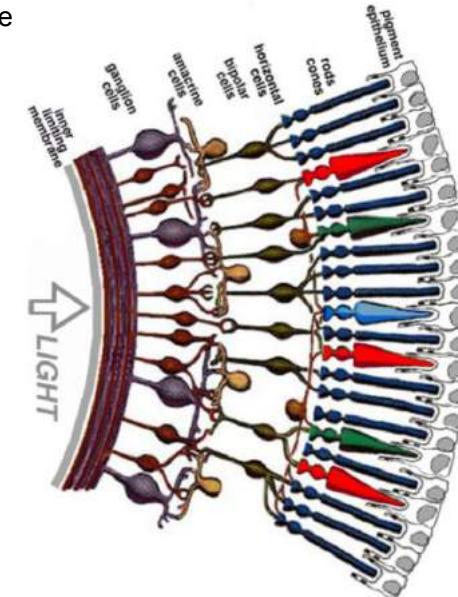
The Retina

The retina contains light sensitive **photoreceptors** (approx 130 million).

These photoreceptors are farthest from the light. But intervening cells are transparent

Photoreceptors **transduce** light to electrical signals (voltage changes).

Transduction = the transformation of one form of energy to another.



Computer Vision / Image Formation (Artificial and Biological)

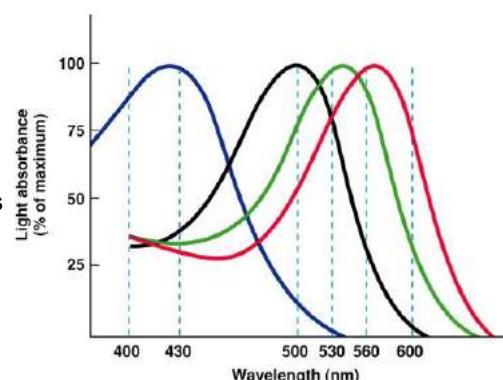
Photoreceptor types

Human eyes have 2 classes of photoreceptor:

- **Rods:**
 - high sensitivity (can operate in dim light)
- **Cones:**
 - low sensitivity (require bright light)
 - 3 sub-types that are sensitive to different wavelengths

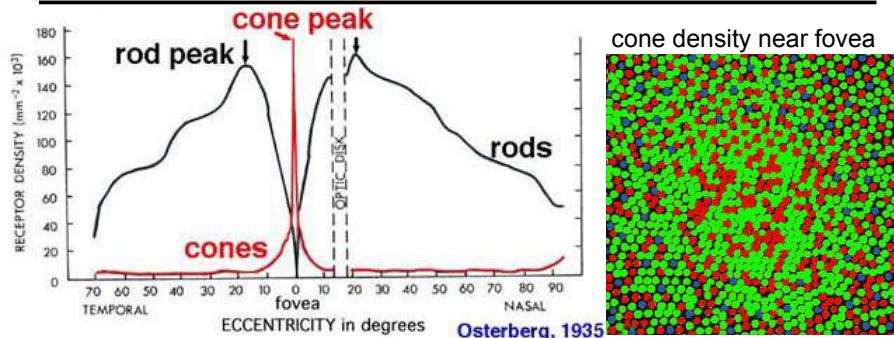
Hence cones provide colour information:

- blue: short-wavelength cones
 - peak sensitivity ~420nm
- green: medium-wavelength cones
 - peak sensitivity ~540nm
- red: long-wavelength cones
 - peak sensitivity ~570nm



Computer Vision / Image Formation (Artificial and Biological)

Distribution of photoreceptors



Photoreceptor types are not evenly distributed across the retina.

- **blind spot:** no photoreceptors (location where optic nerve leaves eye)
- **fovea:** no rods, high density of cones #(blue) << #(red) <= #(green)
- **periphery:** high concentration of rods, few cones

more rods overall

Computer Vision / Image Formation (Artificial and Biological)

Foveal vs peripheral vision

The fovea is small region of high resolution containing mostly cones

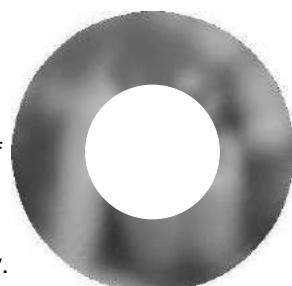
Fovea:

- high resolution (acuity) – due to high density of photoreceptors
- colour – due to photoreceptors being cones
- low sensitivity – due to response characteristics of cones



Periphery:

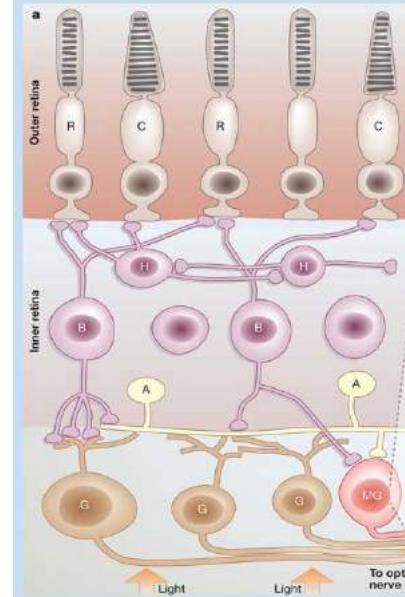
- low resolution (acuity) – due to low density of photoreceptors
- monochrome – due to photoreceptors being rods
- high sensitivity – due to response characteristics of rods



Far more of the brain is devoted to processing information from the fovea than from the periphery.

Computer Vision / Image Formation (Artificial and Biological)

Retinal processing



Computer Vision / Image Formation (Artificial and Biological)

Ganglion cells produce the output of the retina (the axons of all ganglion cells combine to form the optic nerve).

Each eye has approx. 1 million ganglion cells and 100million photoreceptors.

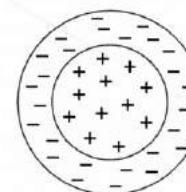
One ganglion cell collects input from multiple photoreceptors:

- Large convergence in periphery
- Smaller convergence in fovea

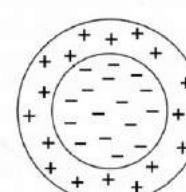
Another reason for the lower acuity in periphery compared to fovea

Ganglion cell responses

ON-center cell



OFF-center cell



Receptive Field Maps

Ganglion cells have centre-surround **receptive fields**.

Receptive Field (RF) = area of visual space (i.e. area of retina) from which a neuron receives input. Or more generally, the properties of a visual stimulus that produces a response from a neuron.

Two types of ganglion cell:

On-centre, off-surround:

- active if central stimulus is brighter than background

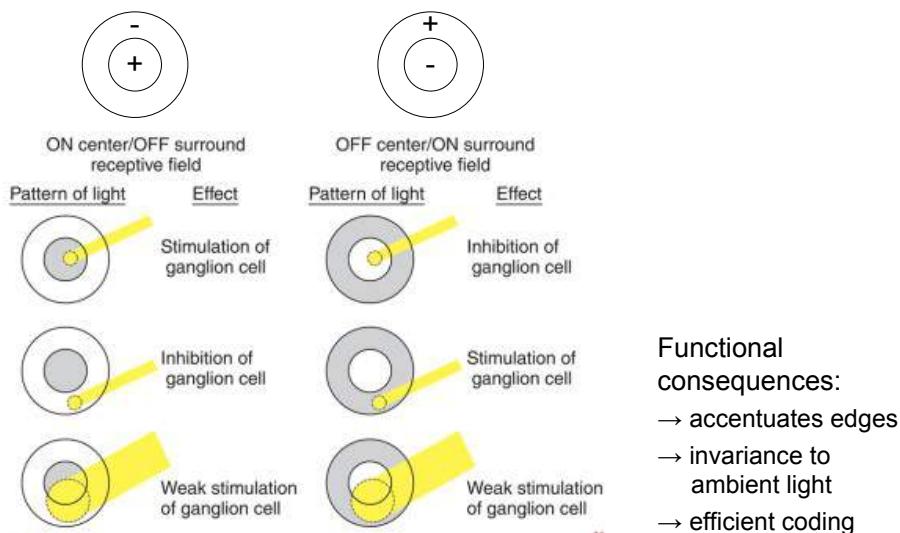
Off-centre, on-surround:

- active if central stimulus is darker than background

Behaviour is generated by ganglion cell being excited (or inhibited) by photoreceptors in the centre of its receptive field and being inhibited (or excited) by photoreceptors surrounding its receptive field.

Computer Vision / Image Formation (Artificial and Biological)

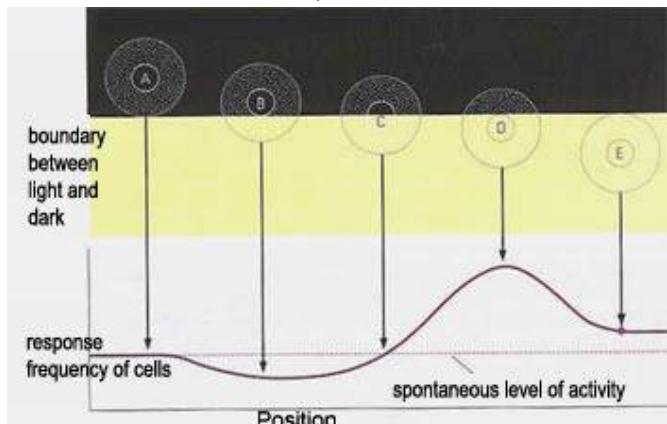
Centre-surround RF function



Computer Vision / Image Formation (Artificial and Biological)

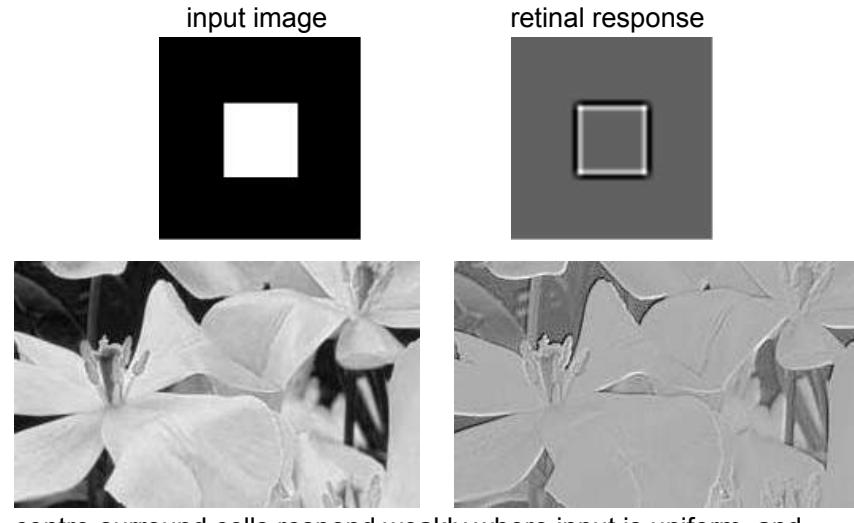
Centre-surround RF: edge enhancement

- A and E: on plane surface input to centre and surround cancels (output at resting – spontaneous - state, greater than zero)
B and D: at contrast discontinuity input to centre and surround unequal (output increased or decreased)



Computer Vision / Image Formation (Artificial and Biological)

Centre-surround RF: edge enhancement



centre-surround cells respond weakly where input is uniform, and strongly where input changes. Hence, strong response near edges.

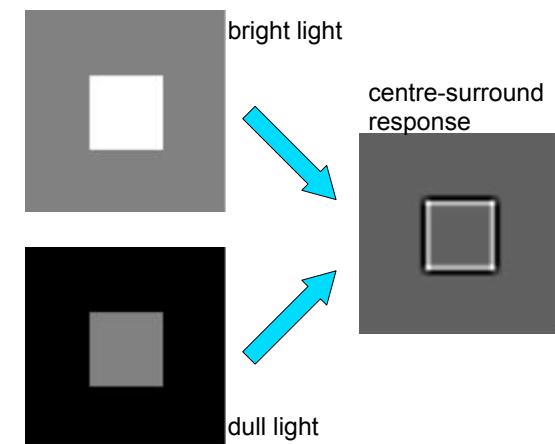
Computer Vision / Image Formation (Artificial and Biological)

Centre-surround RF: invariance to lighting

Ambient lighting conditions, i.e. the **Illuminance (E)**, are generally irrelevant for most perceptual tasks.
e.g. an object should look the same in bright light and dull light.

Centre-surround RFs measure the change in intensity (contrast) between adjacent locations.

This contrast remains constant independent of lighting conditions.



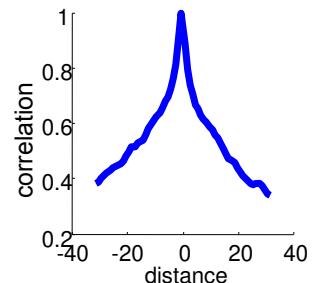
Computer Vision / Image Formation (Artificial and Biological)

Centre-surround RF: efficient coding

Cells with RFs that fall on plane surfaces are only weakly active

Cells with RFs that fall on areas where contrast is changing are strongly active

Natural images have strong spatial correlation (i.e. little intensity change over short distances)



By only signalling where intensity changes, centre-surround RFs:

- minimises neural activity (efficient in terms of energy consumption)
- minimises bandwidth (efficient in terms of information coding) often referred to as "redundancy reduction" or "decorrelation" or "predictive coding"

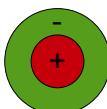
Computer Vision / Image Formation (Artificial and Biological)

Centre-surround RFs: colour opponent cells

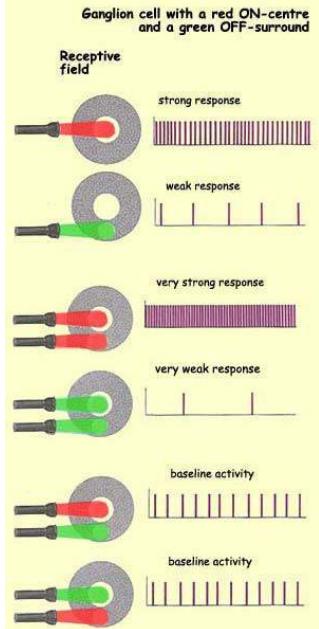
Ganglion cells combine inputs from both rods and cones in a centre-surround configuration.

Input from cones produces colour opponent cells.

e.g. a red on-centre, green off-surround (or red ON/green OFF)

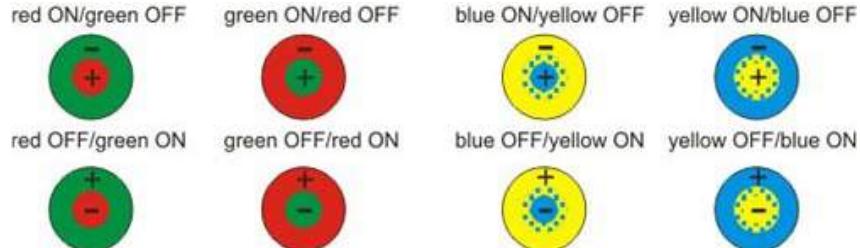


responds most strongly to red light falling within its RFs



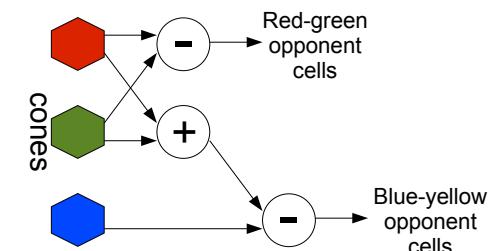
Computer Vision / Image Formation (Artificial and Biological)

Centre-surround RFs: colour opponent cells



A number of different colour combinations occur in the human retina giving rise to a number of different types of colour-opponent cell.

"yellow" is produced by averaging the outputs of red and green cones.

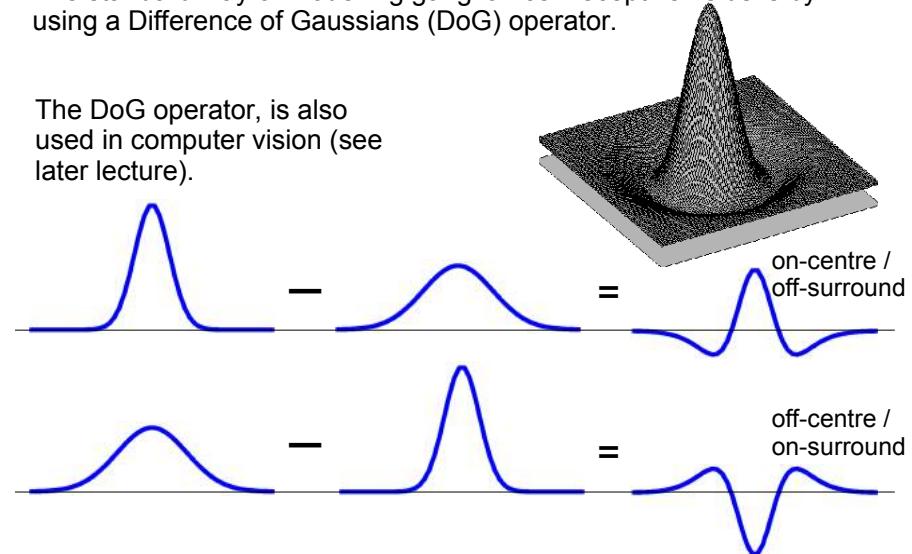


Computer Vision / Image Formation (Artificial and Biological)

Modelling centre-surround RFs

The standard way of modelling ganglion cell receptive fields is by using a Difference of Gaussians (DoG) operator.

The DoG operator, is also used in computer vision (see later lecture).



Computer Vision / Image Formation (Artificial and Biological)

Summary

Image formation is described by the pinhole (perspective) camera model

A point in 3D world projects to a point in 2D image dependent on extrinsic camera parameters, projection operator, intrinsic camera parameters

A lens is required to collect sufficient light to make an image that is both in focus and bright

Light reflected from an object is transduced into a electronic signal by a sensor (CCD array, retinal rods and cones)

The image is sampled at discrete locations (pixels), sampling is uniform in a camera, non-uniform in the retina (periphery vs fovea)

Following image formation the image needs to be analysed:

in biological vision system 1st stage of analysis is performed by centre-surround cells

in artificial vision systems (next week)

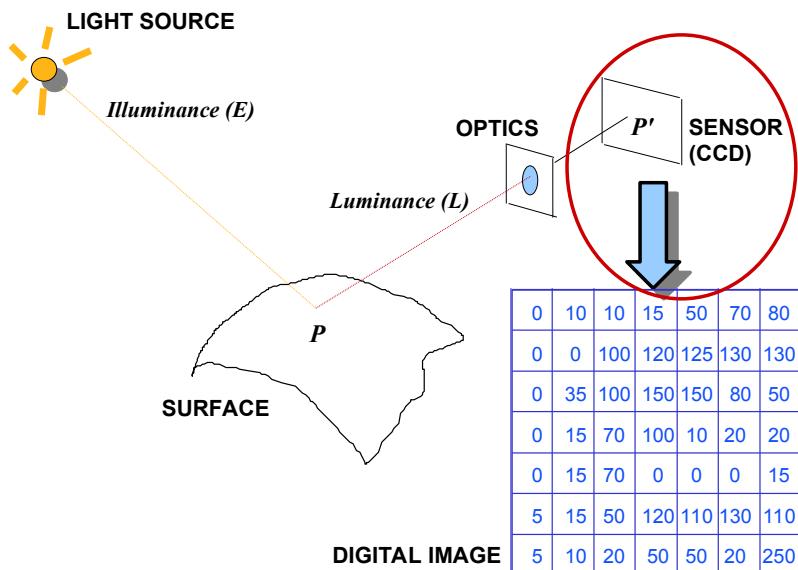
Content

- Camera Sensors

- Charge Coupled Devices (CCDs)
- demosaicing

Computer Vision / Image Formation (Artificial and Biological)

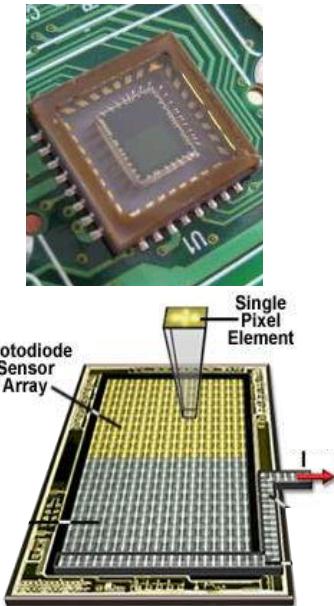
Digital image formation



Computer Vision / Image Formation (Artificial and Biological)

Charge Coupled Device (CCD)

- A semiconductor carrying a two-dimensional matrix of photo-sensors (photo-diodes)
- Each photo-sensor is a small (usually square) electrically isolated capacitive region that can accumulate charge
- Photons incident on a photo-sensor are converted into electrons (via the photoelectric effect)
- The charge accumulated by each element is proportional to the incident light intensity and exposure time
- The pattern of charge across the CCD corresponds to the pattern of incident light intensity (i.e. it is an image)



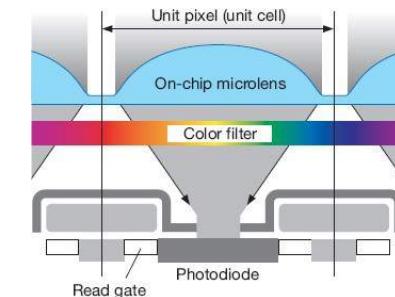
Computer Vision / Image Formation (Artificial and Biological)

Colour from CCD devices

To improve efficiency, microlenses are fabricated on chip to focus incoming light onto each sensor

The photo-sensors in the CCD array are not selective to the wavelength of incoming light.

Colour sensitivity is achieved by adding a colour filter that allows through light from a small band of frequencies associated with a specific colour.



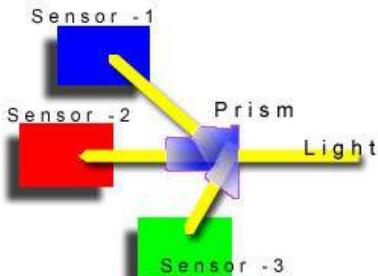
Computer Vision / Image Formation (Artificial and Biological)

Colour from CCD devices

3 CCD solution

A prism splits light into 3 beams of different colour. 3 separate CCD devices record each colour

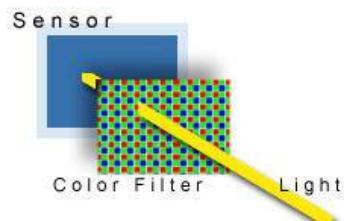
- Expensive, Bulky
- High image quality



1 CCD solution

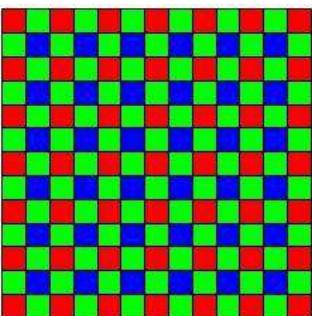
An array of coloured filters is placed over the pixels of single CCD to make different pixels selective to different colours

- Cheap, Compact
- Coarser sampling: lower image quality



Computer Vision / Image Formation (Artificial and Biological)

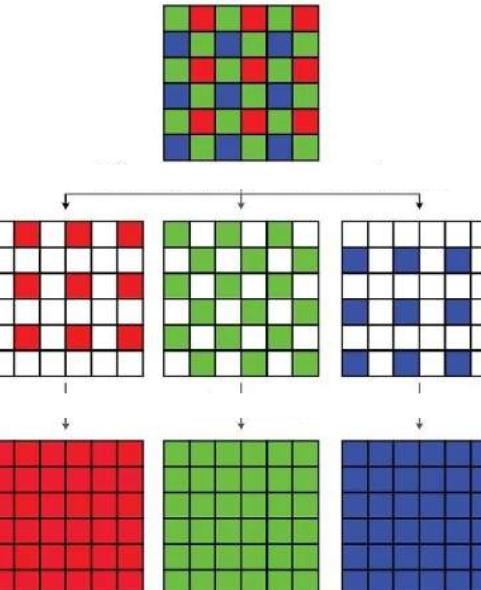
CCD colour masks



The Bayer mask (GRGB) is the most common colour filter used in digital cameras.

There are twice as many green filters as red and blue filters: improves sensitivity to green to which humans are most sensitive.

Demosaicing

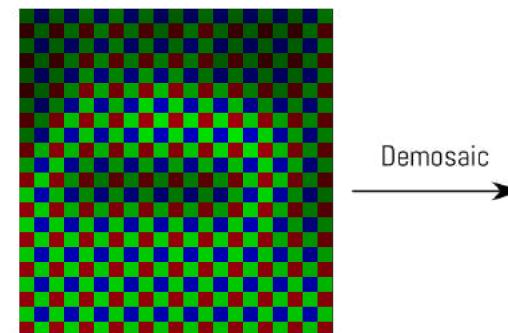


The bayer mask samples colours at specific locations

The result are three colour channels with missing values: individual colour channels are **subsampled**

Demosaicing is the process of filling in the missing values, so that we have R, G and B values at every pixel

Demosaicing



Raw output from Bayer mask image sensor

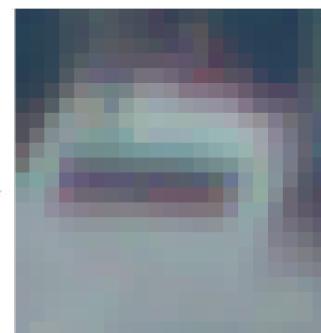


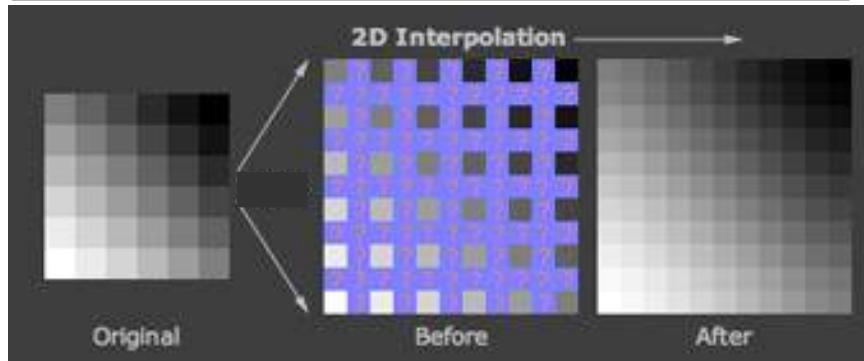
Image after demosaicing

Demosaicing is a process that computes the colour (RGB values) at every pixel based on the local red, green and blue values in the subsampled images.

Computer Vision / Image Formation (Artificial and Biological)

Computer Vision / Image Formation (Artificial and Biological)

Demosaicing and Interpolation



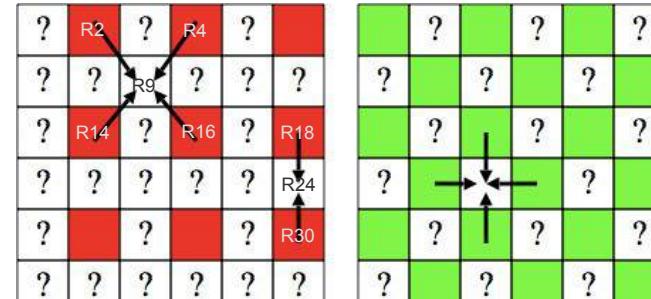
Demosaicing is a method of interpolation

The same interpolation methods can be used when scaling an image

Bilinear Interpolation

Takes average value of nearest two or four pixels from the same colour channel.

- Fast.
- Accurate in smooth regions, inaccurate at edges



$$R9 = \frac{R2 + R4 + R14 + R16}{4}$$

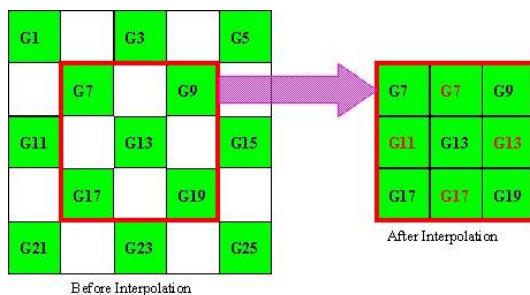
$$R24 = \frac{R18 + R30}{2}$$

Computer Vision / Image Formation (Artificial and Biological)

Nearest Neighbour Interpolation

Copies an adjacent pixel value from the same colour channel.

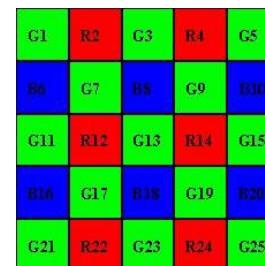
- V. Fast.
- V. Inaccurate.



Smooth Hue Transition Interpolation

Interpolation of green pixels: same as in bilinear interpolation.

Interpolation of red/blue pixels: bilinear interpolation of the ratio ("hue") between red/blue and green.



$$B12 = G12 \times \frac{(B6/G6) + (B8/G8) + (B16/G16) + (B18/G18)}{4}$$

Computer Vision / Image Formation (Artificial and Biological)

Computer Vision / Image Formation (Artificial and Biological)

Edge-Directed Interpolation

The region around a pixel is analysed to determine if a preferred interpolation direction exists

Interpolation performed along axis where change in value is lowest (i.e. not across edges)

G1	R2	G3	R4	G5
B6	G7	B8	G9	B10
G11	R12	G13	R14	G15
B16	G17	B18	G19	B20
G21	R22	G23	R24	G25

e.g. calculating G8

calculate horizontal and vertical gradients:

$$\Delta H = |G7 - G9| \quad \Delta V = |G3 - G13|$$

perform interpolation:

$$\text{if } \Delta H < \Delta V, \quad G8 = (G7 + G9)/2$$

$$\text{else if } \Delta H > \Delta V, \quad G8 = (G3 + G13)/2$$

$$\text{else, } G8 = (G3 + G7 + G9 + G13)/4$$

Interpolation of red/blue pixels: same as
in smooth hue transition interpolation

This week

- Edge and Feature Detection
 - important for subsequent analysis
- Requires Filtering
 - linear filter - replaces each pixel by a linear combination of itself and its neighbours
 - generates a new image
 - low-level vision = image processing
- Requires Convolution
 - the process of applying the filter to the image

Computer Vision / Low-Level Vision (Artificial)

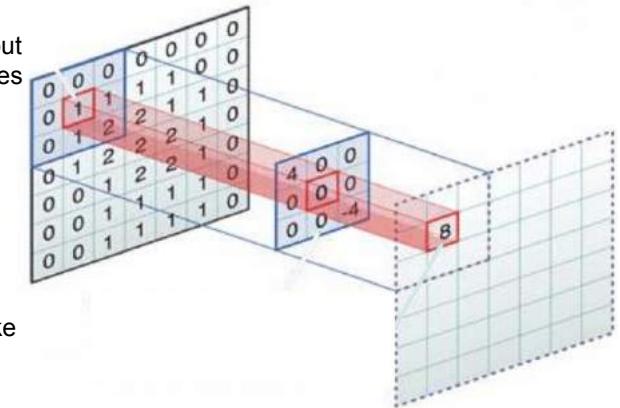
Convolution

$$I'(i, j) = I * H = \sum_{k, l} I(i-k, j-l) H(k, l)$$

To calculate values at each location in the filtered image:

You can imagine sliding the mask across the input image, filling in the values for the output (filtered) image as you go.

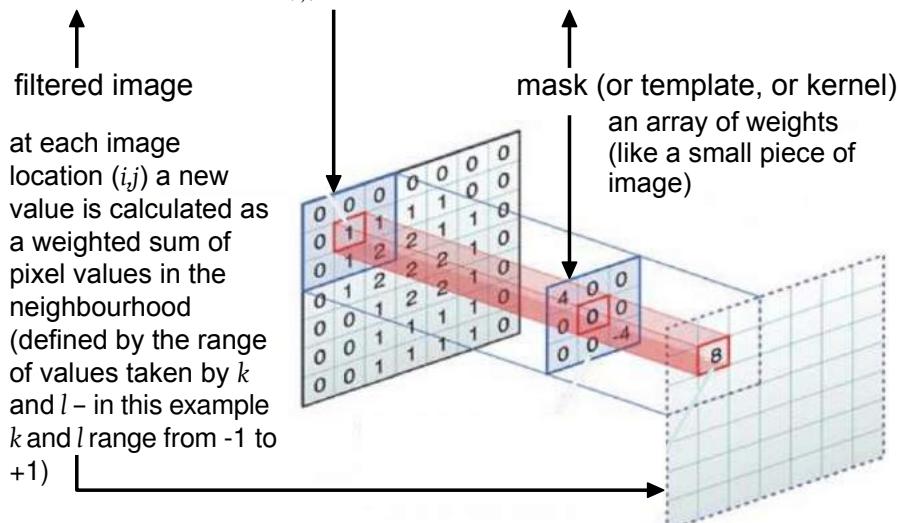
Alternatively, you can imagine the mask replicated at every pixel location in the output image, and the results generated in parallel (like the DoG filters in the retina).



Computer Vision / Low-Level Vision (Artificial)

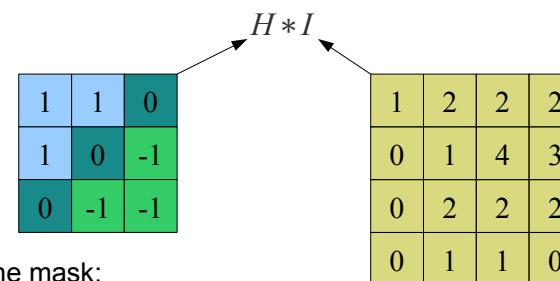
Convolution

$$I'(i, j) = I * H = \sum_{k, l} I(i-k, j-l) H(k, l)$$

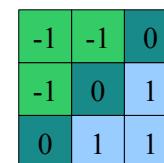


Computer Vision / Low-Level Vision (Artificial)

Convolution: method



Rotate the mask:

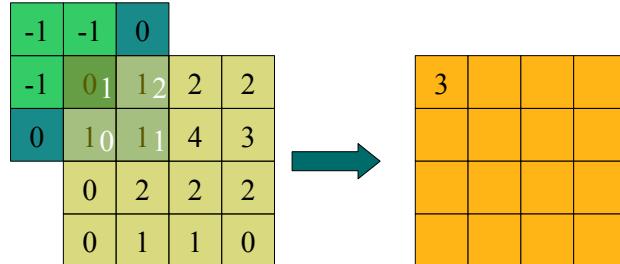


Computer Vision / Low-Level Vision (Artificial)

Convolution: method

For each image pixel in turn:

1. Centre the **rotated** mask over that pixel
2. Multiply each mask element by the corresponding image pixel value
3. Sum these products and write answer in corresponding pixel location in the output image

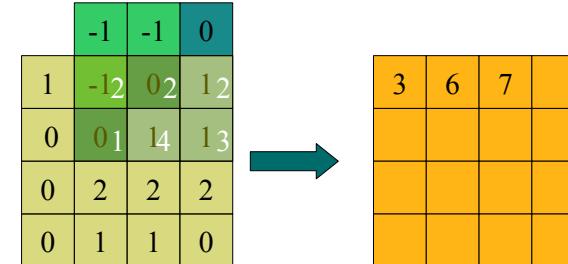


Computer Vision / Low-Level Vision (Artificial)

Convolution: method

For each image pixel in turn:

1. Centre the **rotated** mask over that pixel
2. Multiply each mask element by the corresponding image pixel value
3. Sum these products and write answer in corresponding pixel location in the output image

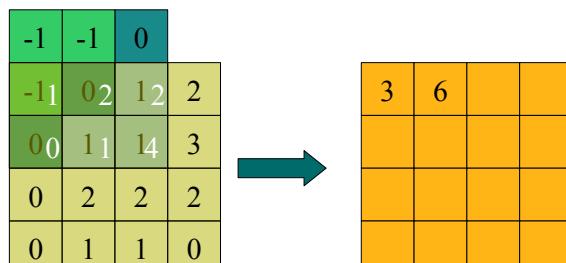


Computer Vision / Low-Level Vision (Artificial)

Convolution: method

For each image pixel in turn:

1. Centre the **rotated** mask over that pixel
2. Multiply each mask element by the corresponding image pixel value
3. Sum these products and write answer in corresponding pixel location in the output image

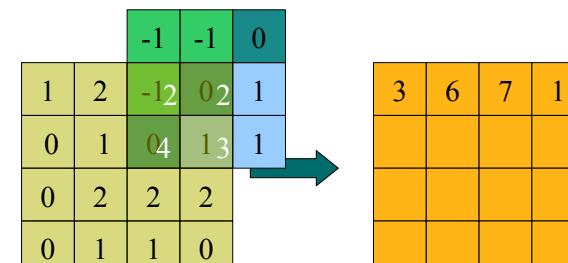


Computer Vision / Low-Level Vision (Artificial)

Convolution: method

For each image pixel in turn:

1. Centre the **rotated** mask over that pixel
2. Multiply each mask element by the corresponding image pixel value
3. Sum these products and write answer in corresponding pixel location in the output image

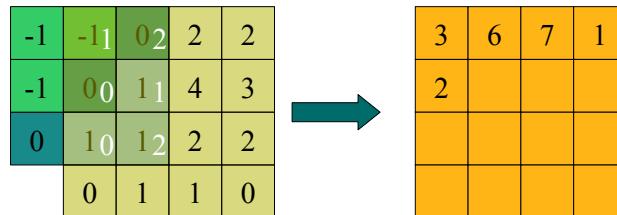


Computer Vision / Low-Level Vision (Artificial)

Convolution: method

For each image pixel in turn:

1. Centre the **rotated** mask over that pixel
2. Multiply each mask element by the corresponding image pixel value
3. Sum these products and write answer in corresponding pixel location in the output image



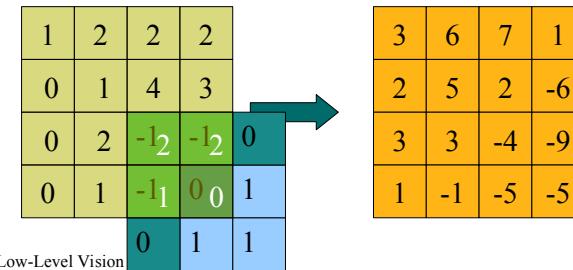
Computer Vision / Low-Level Vision (Artificial)

Convolution: method

For each image pixel in turn:

1. Centre the **rotated** mask over that pixel
2. Multiply each mask element by the corresponding image pixel value
3. Sum these products and write answer in corresponding pixel location in the output image

skipping to the end:

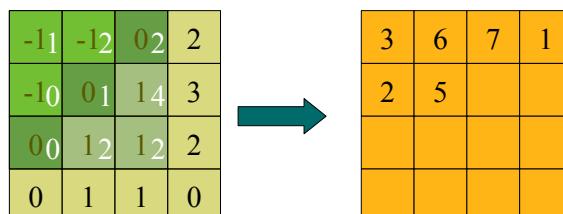


Computer Vision / Low-Level Vision (Artificial)

Convolution: method

For each image pixel in turn:

1. Centre the **rotated** mask over that pixel
2. Multiply each mask element by the corresponding image pixel value
3. Sum these products and write answer in corresponding pixel location in the output image



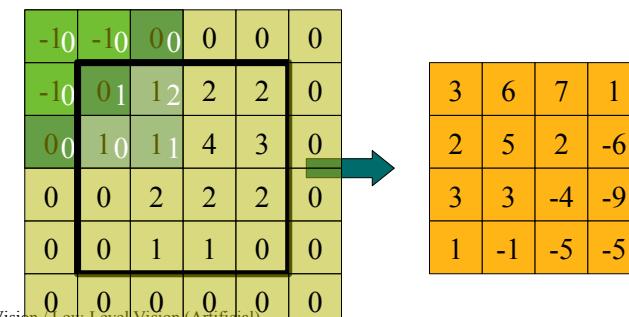
Computer Vision / Low-Level Vision (Artificial)

Convolution at image boundaries

How to deal with image boundaries (where mask “falls off” image)?

Most common methods:

1. pad the input image with zeros (area outside the black square in example below). i.e. the implicit assumption made in the preceding example.
2. make the output image smaller than the input image (red area in example below). i.e. only apply mask at locations on the input image where it does not fall off the input image.



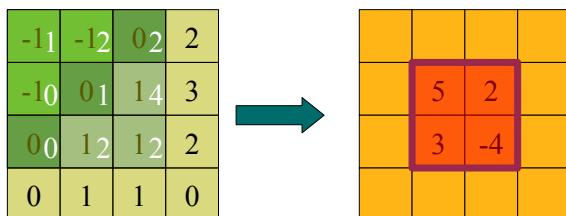
Computer Vision / Low-Level Vision (Artificial)

Convolution at image boundaries

How to deal with image boundaries (where mask “falls off” image)?

Most common methods:

1. pad the input image with zeros (area outside the black square in example below). i.e. the implicit assumption made in the preceding example.
2. make the output image smaller than the input image (red area in example below). i.e. only apply mask at locations on the input image where it does not fall off the input image.



Computer Vision / Low-Level Vision (Artificial)

Convolution and Correlation

convolution:

$$\begin{aligned}I'(i, j) &= I * H = \sum_{k,l} I(i-k, j-l) H(k, l) \\&= \sum_{k,l} I(i+k, j+l) \underbrace{H(-k, -l)}_{\text{rotated mask}}\end{aligned}$$

cross-correlation:

$$I'(i, j) = I \star H = \sum_{k,l} I(i+k, j+l) H(k, l)$$

Hence, if mask is not rotated the result will be the cross-correlation rather than the convolution of I and H .

If mask is symmetrical (i.e. $H(k, l) = H(-k, -l)$) then cross-correlation = convolution.

Computer Vision / Low-Level Vision (Artificial)

Convolution and Correlation

convolution is:

- commutative $I * H = H * I$
- associative $(I * H) * G = I * (H * G)$

cross-correlation is **not**:

- commutative $I \star H \neq H \star I$
- associative $(I \star H) \star G \neq I \star (H \star G)$

For this reason convolution is used in preference to cross-correlation (despite the inconvenience of needing to rotate the mask)

Computer Vision / Low-Level Vision (Artificial)

Separable Masks

A 2D convolution is **separable** if the kernel H can be written as a convolution of two (row) vectors

$$\text{i.e. if } H = h_1 * h_2^T$$

A separable convolution can be performed as two 1D convolutions

$$\begin{aligned}I * H &= I * (h_1 * h_2^T) \\&= (I * h_1) * h_2^T\end{aligned}$$

Note $h_1 * h_2^T = h_2^T \times h_1$ (i.e. the convolution of two vectors equals the product of those vectors)

Separable convolutions are much more efficient to compute.
Convolving an image with an $m \times n$ pixel kernel takes:

$m \times n$ products per pixel for 2D mask

$m+n$ products per pixel for two 1D masks

Computer Vision / Low-Level Vision (Artificial)

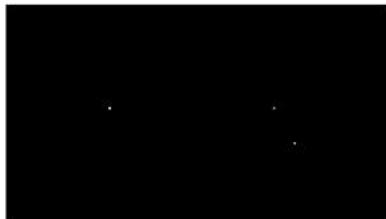
Masks as point-spread functions

Convolve a mask with a simple image that has just an isolated white dot on a black background.

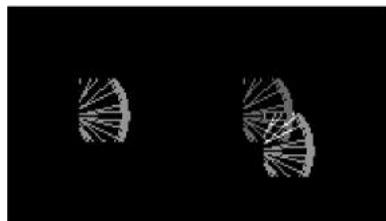
The output will be the mask itself shifted by the row and column numbers of the isolated pixel in the input.

An ordinary image can be thought of as a combination of such points - one for each pixel. So the result of a convolution can be thought of as a superimposition of masks, each one weighted by the intensity of an image pixel.

Input Image



Output Image



$$\text{Mask} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Computer Vision / Low-Level Vision (Artificial)

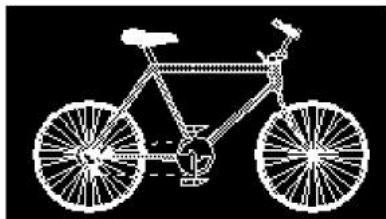
Masks as templates

The convolution output is a maximum when large values in the input get multiplied by large values in the mask.

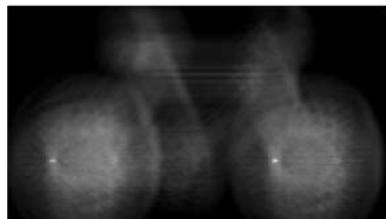
This means that convolution masks respond most strongly to image features that resemble the rotated mask.

The rotated mask is like a template which is scanned across the image to find image features that match that template.

Input Image



Output Image



$$\text{Mask} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Computer Vision / Low-Level Vision (Artificial)

Mask weights values

The values chosen for the mask determine the effects of the convolution.

In theory, we could choose any values for the weights.

In practice, two categories of mask are commonly used:

- smoothing masks
- differencing masks

Computer Vision / Low-Level Vision (Artificial)

Mask weight values

The weights in a smoothing mask add up to 1 to preserve average grey levels.

e.g.	1/9	1/9	1/9
	1/9	1/9	1/9
	1/9	1/9	1/9

The weights in differencing masks add up to 0 to generate a zero response to constant image regions.

e.g.	-1	-1	-1
	-1	8	-1
	-1	-1	-1

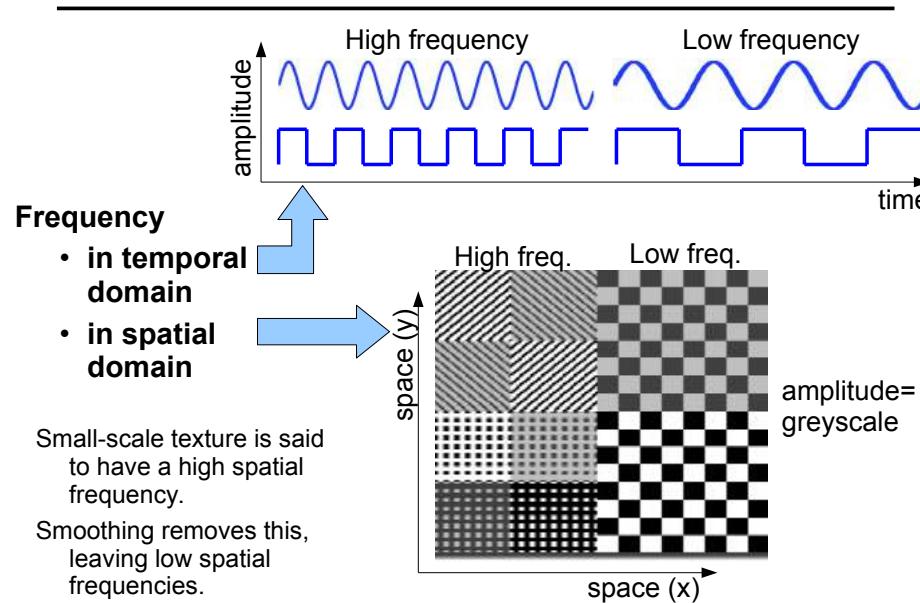
Computer Vision / Low-Level Vision (Artificial)

Content

- **Smoothing Images**
 - spatial frequency
 - box masks
 - Gaussian masks

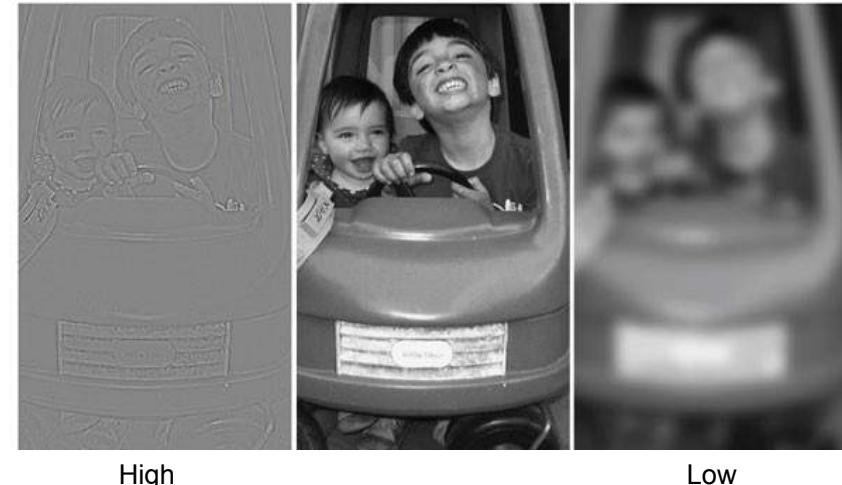
Computer Vision / Image Formation (Artificial and Biological)

Spatial Frequency



Computer Vision / Low-Level Vision (Artificial)

Spatial Frequency: example



Computer Vision / Low-Level Vision (Artificial)

Box mask

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

Each pixel replaced by average of itself and its eight neighbours.

Generates a smoothed (blurred) image.

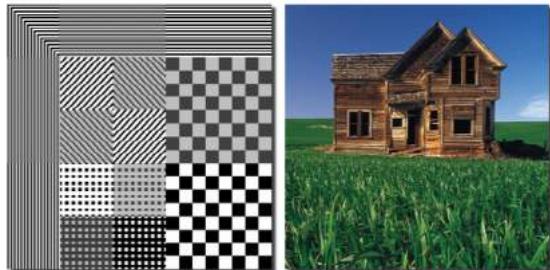
Useful

Called “mean filter” or “box mask”.

Computer Vision / Low-Level Vision (Artificial)

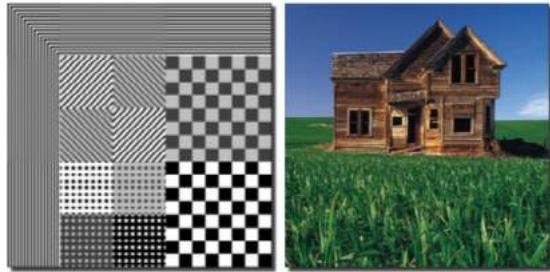
Smoothing mask example

Original Images



Images convolved
with 3x3 box mask

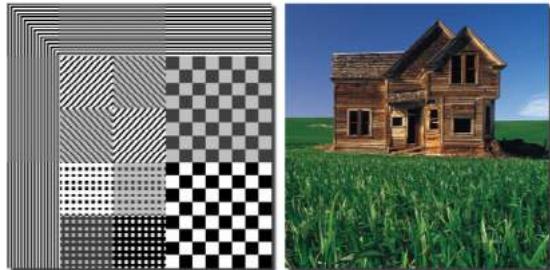
1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



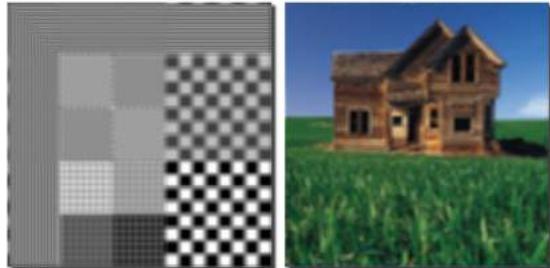
Computer Vision / Low-Level Vision (Artificial)

Smoothing mask example

Original Images



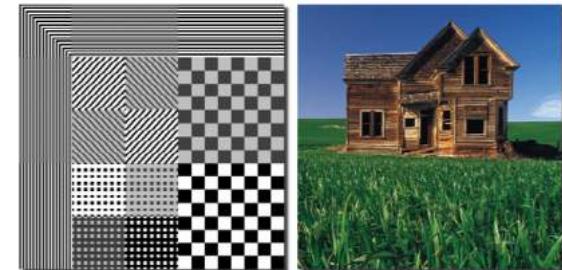
Images convolved
with 9x9 box mask



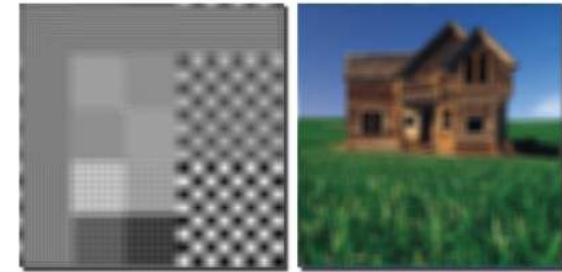
Computer Vision / Low-Level Vision (Artificial)

Smoothing mask example

Original Images



Images convolved
with 17x17 box
mask

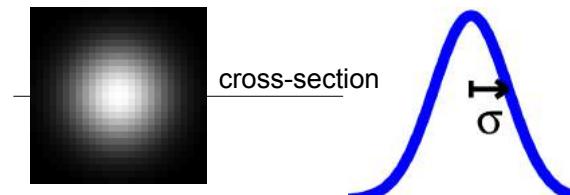


Computer Vision / Low-Level Vision (Artificial)

Gaussian mask

The box mask is not very good at smoothing as it has sharp edges.

The Gaussian mask is more effective as it falls off gently at the edges. It gives more weight to nearby pixels.

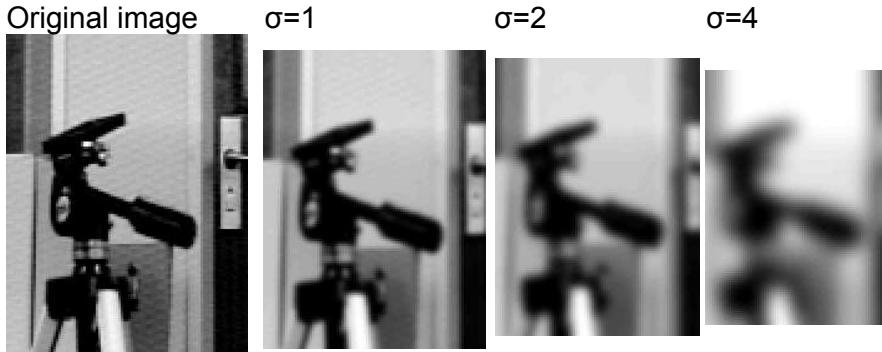


$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right)$$

The Gaussian mask effectively removes any spatial frequencies with a period or wavelength smaller than the mask dimensions (defined by σ = standard deviation, or “scale”).

Computer Vision / Low-Level Vision (Artificial)

Smoothing with Gaussian mask



“Scale” = the standard deviation of the Gaussian (i.e. σ)

as this parameter goes up:

- the size of the mask needs to increase (mask width should be $\geq 6\sigma$)
- more pixels are involved in the average
- the image gets more blurred (more high frequencies suppressed)
- noise is more effectively suppressed

Computer Vision / Low-Level Vision (Artificial)

Separability of the Gaussian mask

$$\begin{aligned} G(x, y) &= \frac{1}{2\pi\sigma^2} \exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right) \\ &= \left\langle \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-x^2}{2\sigma^2}\right) \right\rangle \left\langle \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-y^2}{2\sigma^2}\right) \right\rangle \end{aligned}$$

i.e. 2D Gaussian is the product of two 1D Gaussians.

Each 1D Gaussian is a function of one variable only (either x or y)

Computer Vision / Low-Level Vision (Artificial)

Content

• Differencing Masks

- 1st derivative masks
- 2nd derivative masks
- Laplacian mask

Computer Vision / Image Formation (Artificial and Biological)

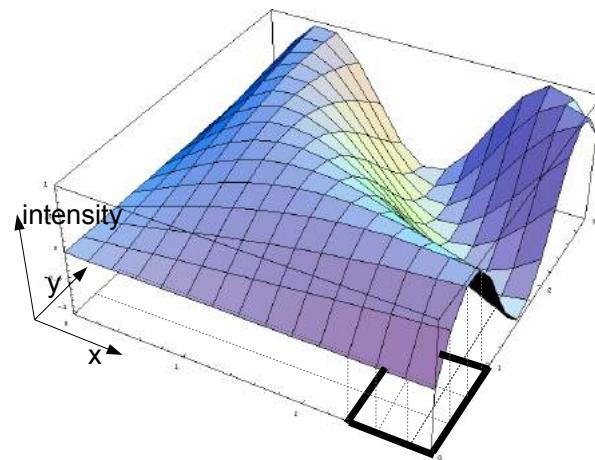
Difference Masks

As well as calculating averages (as with box and Gaussian masks), convolution can also be used to calculate differences.

The difference between pixel values measures the gradient of the intensity values.

Hence:

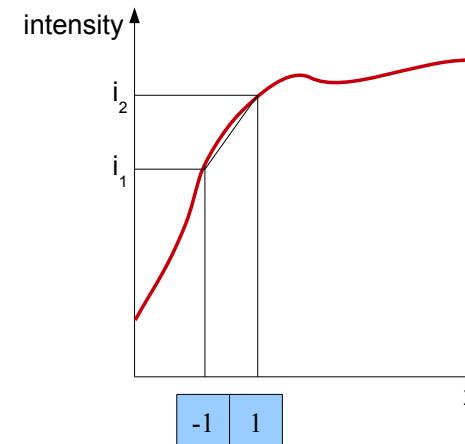
- smoothing masks approximate integration
- difference masks approximate differentiation



Computer Vision / Low-Level Vision (Artificial)

1st derivative mask

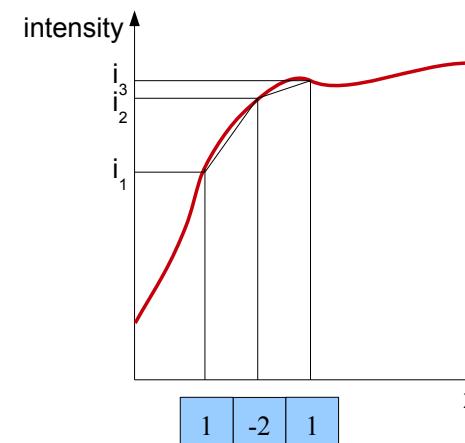
Estimate of gradient is $\frac{\Delta y}{\Delta x}$ i.e.: $(i_2 - i_1)$



Computer Vision / Low-Level Vision (Artificial)

2nd derivative mask

Estimate of change of gradient is $(i_3 - i_2) - (i_2 - i_1)$



Computer Vision / Low-Level Vision (Artificial)

Difference masks for different directions

vertical horizontal diagonals ← **Mask orientation**
 horizontal vertical diagonals ← **Orientation of intensity change detected**

1st derivative masks:

$$\begin{array}{|c|c|} \hline -1 & \\ \hline 1 & \\ \hline \end{array} \approx -\delta/\delta y \quad \begin{array}{|c|c|} \hline -1 & 1 \\ \hline & \\ \hline \end{array} \approx -\delta/\delta x$$

$$\begin{array}{|c|c|} \hline -1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline 0 & -1 \\ \hline 1 & 0 \\ \hline \end{array}$$

2nd derivative masks:

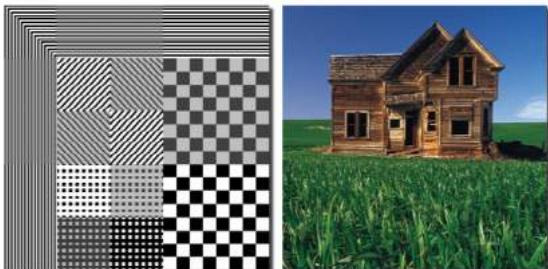
$$\begin{array}{|c|c|c|} \hline -1 & & \\ \hline 2 & & \\ \hline -1 & & \\ \hline \end{array} \approx -\delta^2/\delta y^2 \quad \begin{array}{|c|c|c|} \hline -1 & 2 & -1 \\ \hline & & \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & -1 \\ \hline \end{array} \approx -\delta^2/\delta x^2$$

$$\begin{array}{|c|c|c|} \hline -1 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & -1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|} \hline 0 & 0 & -1 \\ \hline 0 & 2 & 0 \\ \hline -1 & 0 & 0 \\ \hline \end{array}$$

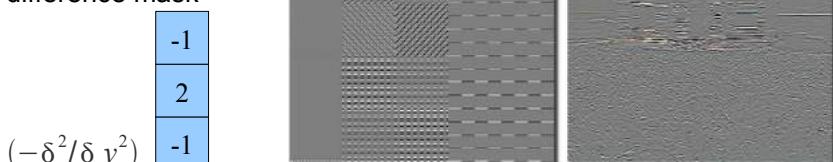
Computer Vision / Low-Level Vision (Artificial)

Difference mask example

Original Images



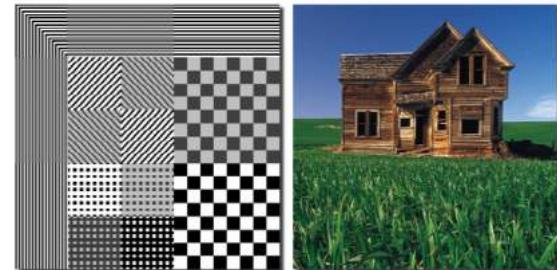
Images convolved with vertical difference mask



Computer Vision / Low-Level Vision (Artificial)

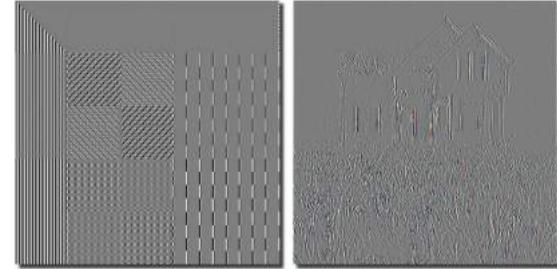
Difference mask example

Original Images



Images convolved with horizontal difference mask

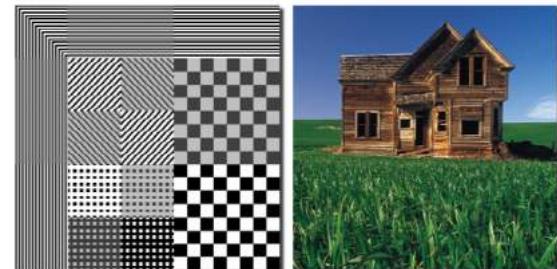
$$\begin{array}{|c|c|c|} \hline -1 & 2 & -1 \\ \hline & & \\ \hline \end{array} \quad (-\delta^2/\delta x^2)$$



Computer Vision / Low-Level Vision (Artificial)

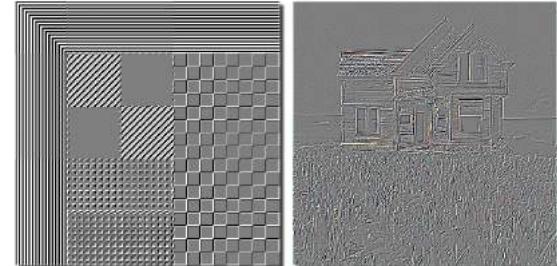
Difference mask example

Original Images



Images convolved with diagonal difference mask

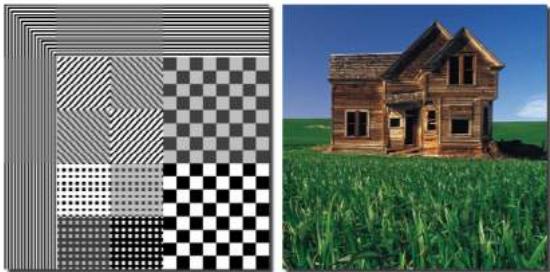
$$\begin{array}{|c|c|c|} \hline -1 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & -1 \\ \hline \end{array}$$



Computer Vision / Low-Level Vision (Artificial)

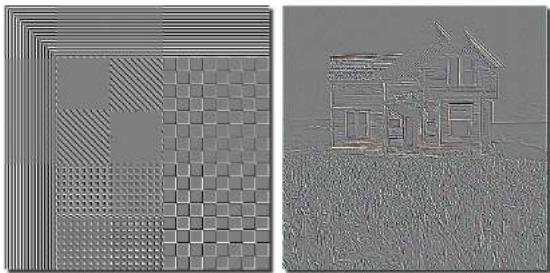
Difference mask example

Original Images



Images convolved
with diagonal
difference mask

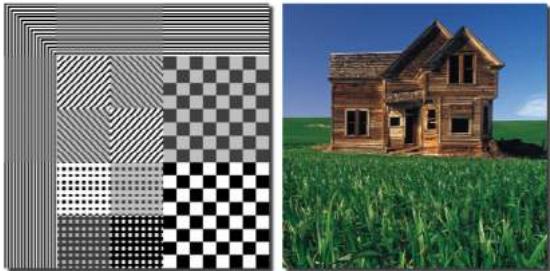
0	0	-1
0	2	0
-1	0	0



Computer Vision / Low-Level Vision (Artificial)

Difference mask example

Original Images



Images convolved with
vertical + horizontal +
both diagonal
difference mask

-1	-1	-1
-1	8	-1
-1	-1	-1

$$\approx -\delta^2/\delta x^2 - \delta^2/\delta y^2$$

Computer Vision / Low-Level Vision (Artificial)

The Laplacian mask

The final example be seen as a combination of 2nd derivative difference masks in each direction.

It therefore detects intensity discontinuities at all orientations.

-1	-1	-1
-1	8	-1
-1	-1	-1

$$\approx -\delta^2/\delta x^2 - \delta^2/\delta y^2$$

Called the Laplacian mask.

However, note that strictly the Laplacian should be the additive inverse of this mask.

Computer Vision / Low-Level Vision (Artificial)

Content

• Edge Detection

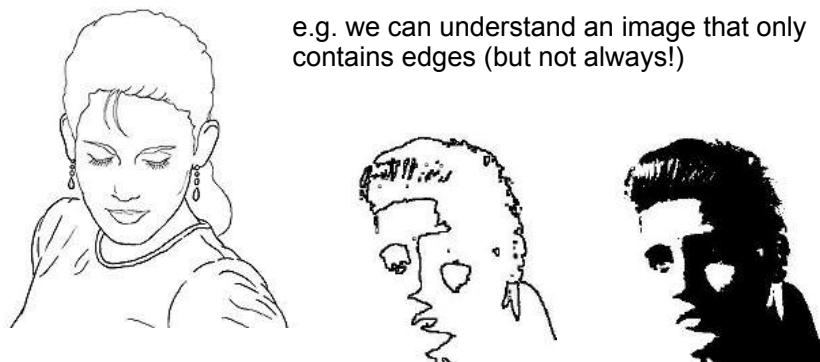
- Intensity discontinuities
- Laplacian of Gaussian / DoG mask
- Gaussian derivative masks
- The Canny edge detector

Computer Vision / Image Formation (Artificial and Biological)

Intensity discontinuities

Discontinuities in the intensity (pixel) values of an image often correspond to useful (meaningful) physical characteristics, such as the boundaries of objects.

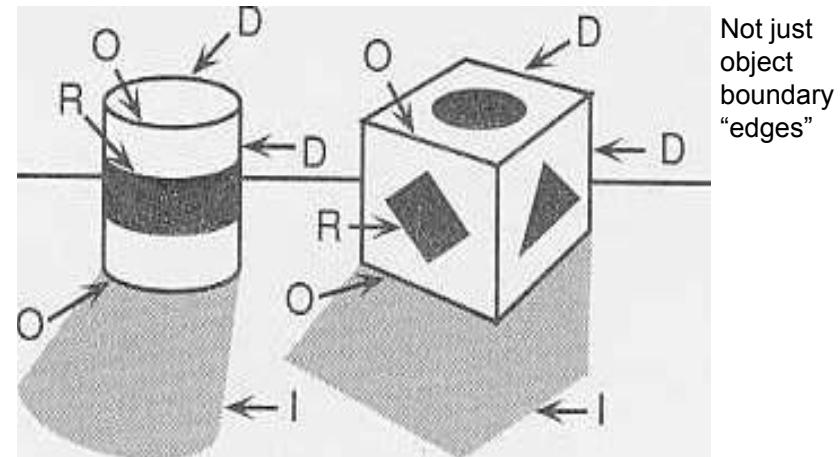
These features are useful, and can be sufficient, for recognising the content of an image.



e.g. we can understand an image that only contains edges (but not always!)

Computer Vision / Low-Level Vision (Artificial)

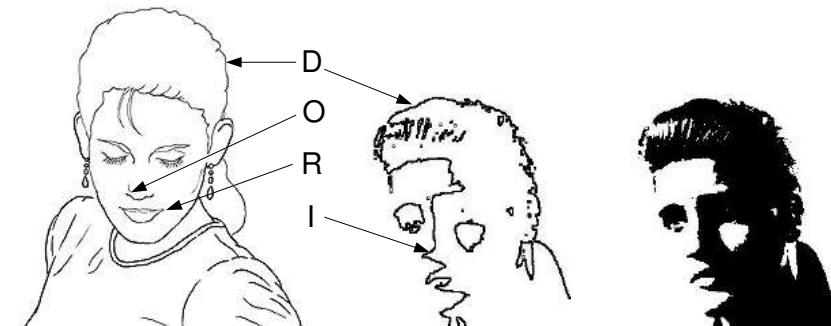
Causes of intensity discontinuities



Depth discontinuity (D) – due to surfaces at different distances
Orientation discontinuity (O) – due to changes in the orientation of a surface
Reflectance discontinuity (R) – due to change in surface material properties
Illumination discontinuity (I) – e.g. shadow boundaries

Computer Vision / Low-Level Vision (Artificial)

Causes of intensity discontinuities



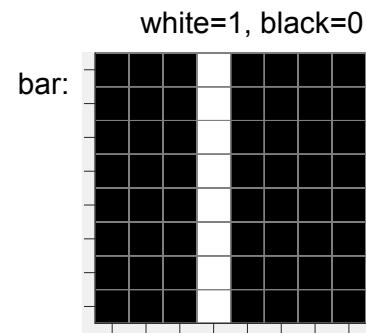
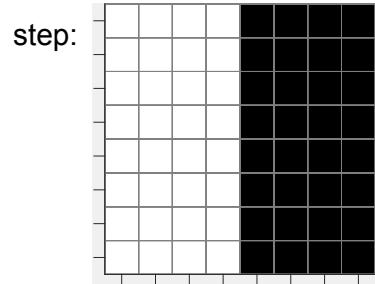
Depth discontinuity (D) – due to surfaces at different distances
Orientation discontinuity (O) – due to changes in the orientation of a surface
Reflectance discontinuity (R) – due to change in surface material properties
Illumination discontinuity (I) – e.g. shadow boundaries: **not helpful for object recognition**

Computer Vision / Low-Level Vision (Artificial)

Edge detection

An edge is an image region in which the intensity gradients have a common direction and large magnitudes (in a direction orthogonal to the edge).

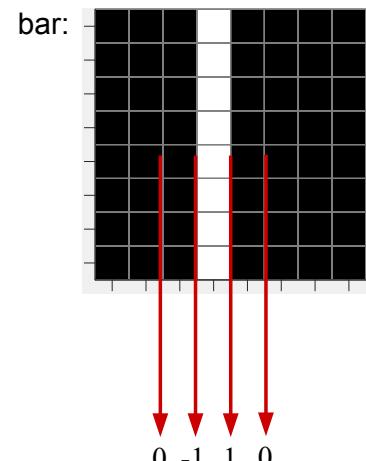
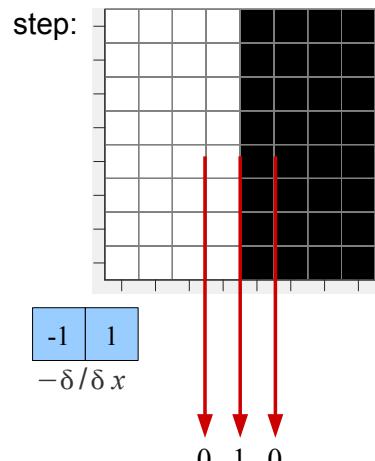
Idealised Edge Types:



Computer Vision / Low-Level Vision (Artificial)

Edge detection: Idealised Edges

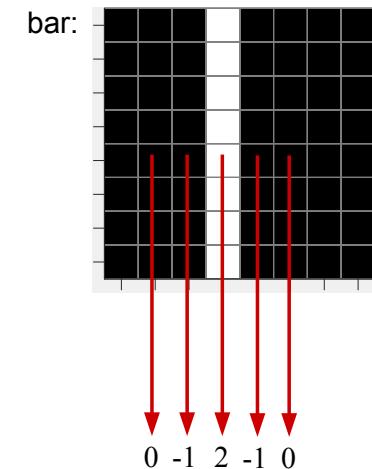
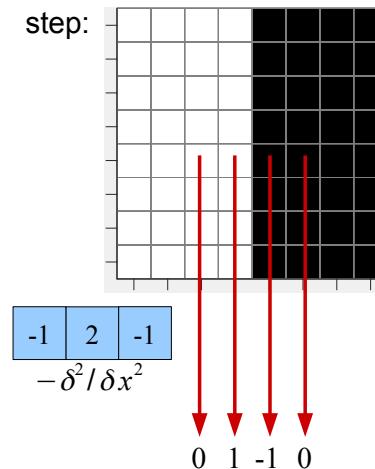
The difference masks considered previously will detect these idealised edges.



Computer Vision / Low-Level Vision (Artificial)

Edge detection: Idealised Edges

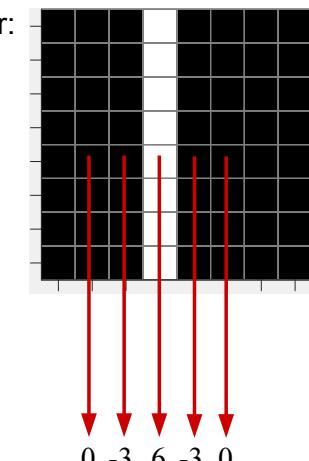
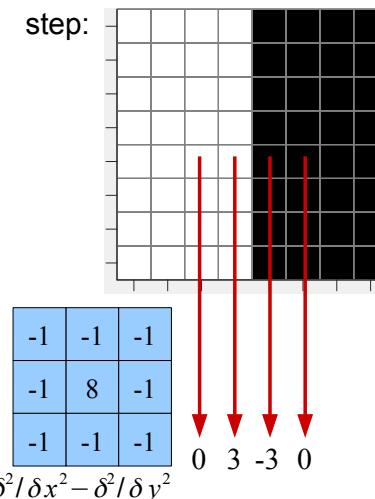
The difference masks considered previously will detect these idealised edges.



Computer Vision / Low-Level Vision (Artificial)

Edge detection: Idealised Edges

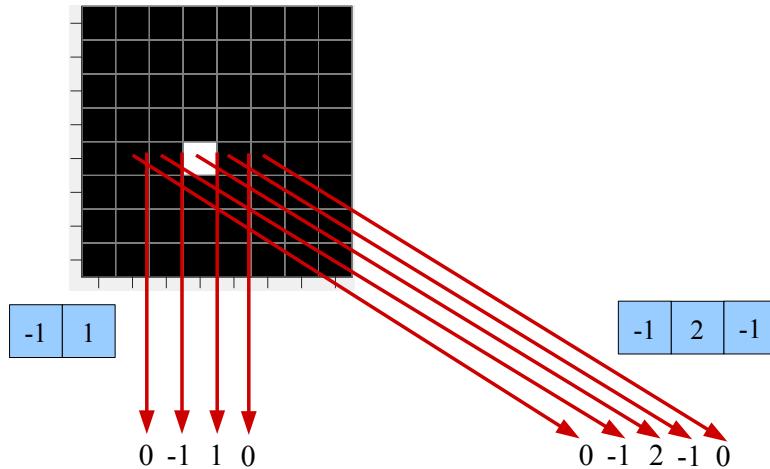
The difference masks considered previously will detect these idealised edges.



Computer Vision / Low-Level Vision (Artificial)

Edge detection: Idealised Edges

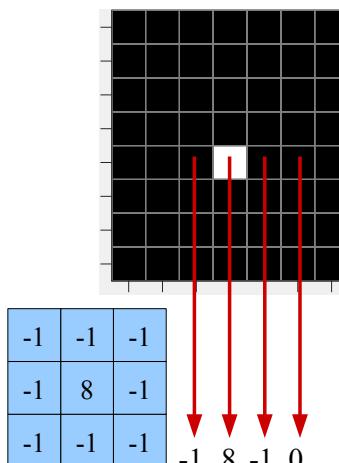
However, difference masks are also sensitive to a single high intensity dot, one pixel in size (i.e. noise).



Computer Vision / Low-Level Vision (Artificial)

Edge detection: Idealised Edges

However, difference masks are also sensitive to a single high intensity dot, one pixel in size (i.e. noise).



Computer Vision / Low-Level Vision (Artificial)

Edge detection

Convolving with a differencing mask:

- enhances edges
- but also enhances noise

Convolving with a smoothing mask:

- removes noise
- but also blurs edges

There is a trade-off between edge detection and noise suppression.

Edge detection requires:

- a spatial scale, established using a smoothing operator (a Gaussian mask)
- a differencing operator to find significant grey-level changes (at that spatial scale)

Computer Vision / Low-Level Vision (Artificial)

Laplacian of Gaussian / DoG mask

A standard approach is to combine Gaussian smoothing with a Laplacian operator.

The two masks can be combined by convolution into a single mask.

This is called the Laplacian of Gaussian (LoG) mask.

$$\begin{array}{c} \text{Gaussian Mask} \\ \times \begin{bmatrix} -1/8 & -1/8 & -1/8 \\ -1/8 & 1 & -1/8 \\ -1/8 & -1/8 & -1/8 \end{bmatrix} = \text{LoG Mask} \end{array} \approx -\frac{\delta^2}{\delta x^2} G_\sigma - \frac{\delta^2}{\delta y^2} G_\sigma$$

It is similar to the Difference of Gaussians or DoG mask (or “centre-surround” or “Mexican hat”).

$$\begin{array}{c} G_{\sigma 1} \\ - \quad G_{\sigma 2} = \text{DoG Mask} \end{array}$$

Computer Vision / Low-Level Vision (Artificial)

Gaussian derivative masks

Alternatively, the 1st derivative difference operators can be combined with Gaussian smoothing. These tend to produce more robust results than the DoG mask.

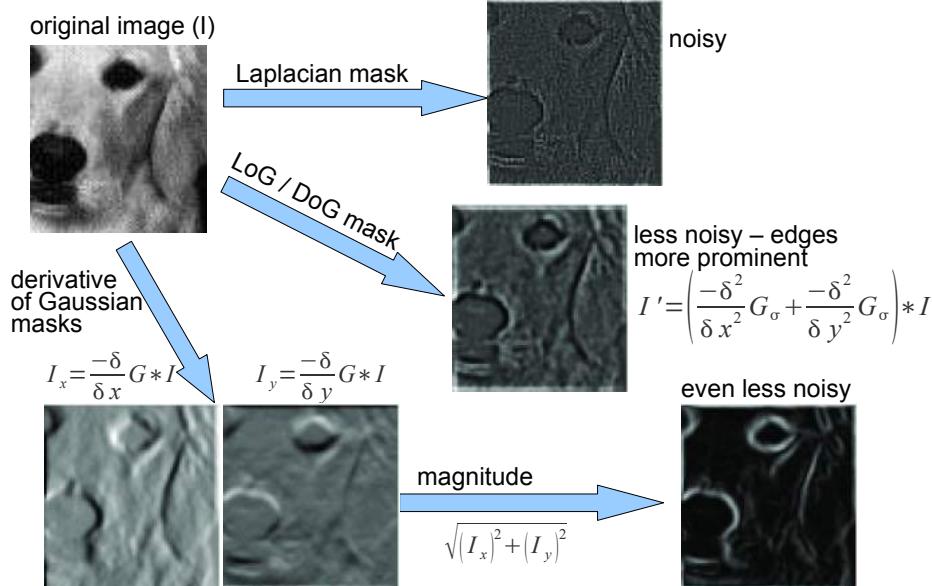
$$\text{original image} * \begin{bmatrix} -1 & 1 \end{bmatrix} = \frac{-\delta}{\delta x} G_\sigma \text{ x gradient (at scale } \sigma\text{)}$$

$$\text{original image} * \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{-\delta}{\delta y} G_\sigma \text{ y gradient (at scale } \sigma\text{)}$$

Such masks emphasise vertical and horizontal structure at the scale set by the σ parameter of the Gaussian component.

Computer Vision / Low-Level Vision (Artificial)

Edge detection comparison

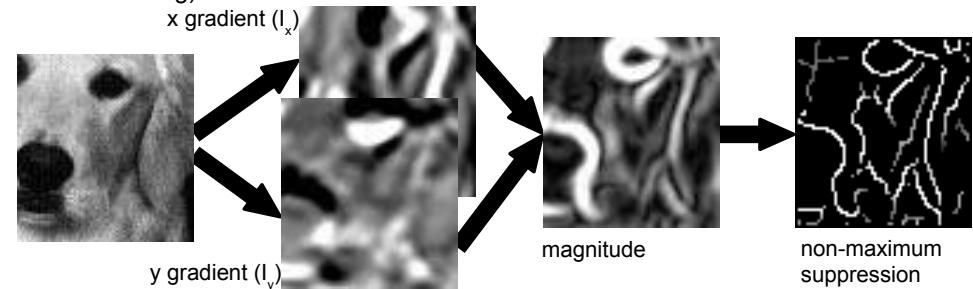


Computer Vision / Low-Level Vision (Artificial)

The Canny edge detector

One of the most popular edge detectors is based on Gaussian derivative masks

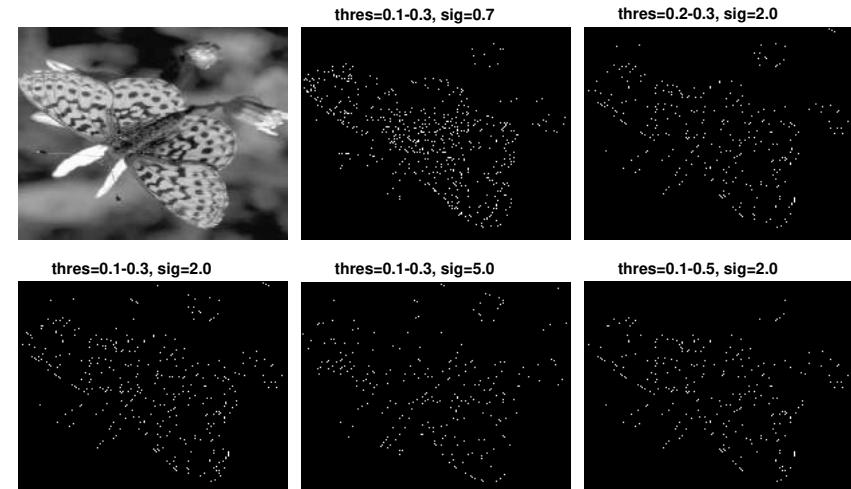
1. convolution with derivatives of Gaussian masks
2. calculation of magnitude $M = \sqrt{(I_x)^2 + (I_y)^2}$ and direction $D = \arctan(I_y/I_x)$
3. non-maximum suppression (thin multi-pixel wide “ridges” down to a single pixel by removing current pixel if neighbouring pixels perpendicular to the direction of the edge have a higher magnitude)
4. recursive hysteresis thresholding (accept/reject pixels above/below high/low threshold, others accepted if neighbouring an edge or neighbouring a pixel with value between thresholds that becomes an edge due to recursive hysteresis thresholding)



Computer Vision / Low-Level Vision (Artificial)

Edge detection: issues

Results of edge detection are highly dependent on the parameter values, e.g.:



Computer Vision / Low-Level Vision (Artificial)

Edge detection: issues

Parameters that work well for one image, may be poor for another, e.g.:



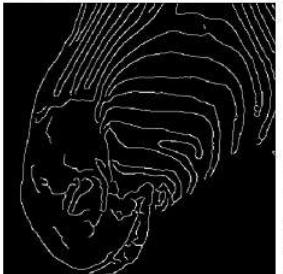
thres=0.1-0.2, sig=3.0



thres=0.1-0.2, sig=3.0

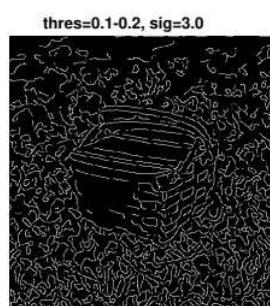
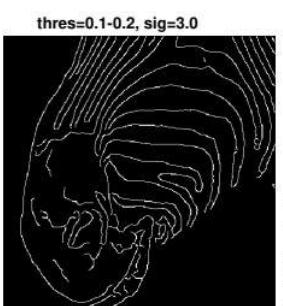


thres=0.1-0.2, sig=3.0



Edge detection: issues

Not all detected edges corresponding to meaningful/useful features in the image, such as the contours of objects



Content

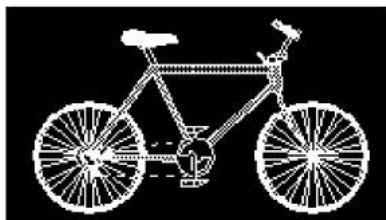
- **Multi-Scale Feature Detection**
 - Image Pyramids
 - Gaussian Image Pyramids
 - Laplacian Image Pyramids

Computer Vision / Image Formation (Artificial and Biological)

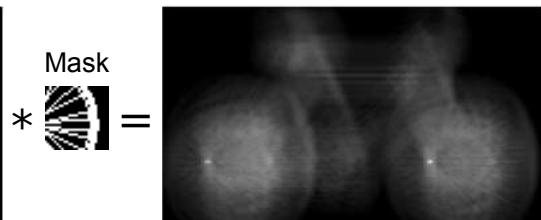
General algorithm for feature detection

Recall that a mask can be viewed as a template for a particular image feature (one identical to the rotated mask).

Input Image



Output Image



- Convolve the image with a mask for the feature.
- Choose a threshold.
- Detect the feature in those parts of the image where the convolved image exceeds the threshold.

Computer Vision / Low-Level Vision (Artificial)

Template matching across scales

Convolution scans the template across the image to find locations where the image matches the template

However, we don't know the scale of the feature we are trying to find, as this will depend on the unknown parameters of the image formation process



How to find image features with invariance to scale?

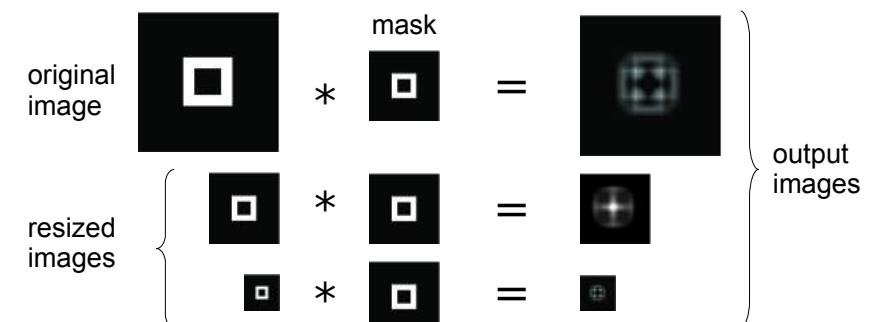
Computer Vision / Low-Level Vision (Artificial)

Image Pyramid

To find image features with invariance to scale, we could:

1. apply filters of different sizes to the image, or
2. apply filters of a fixed size to the image presented at different sizes

The 2nd method is usually used. This is called an image pyramid.



Computer Vision / Low-Level Vision (Artificial)

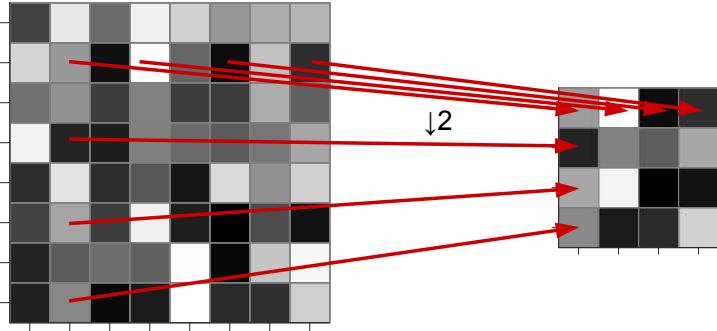
Down-sampling

Decreasing the size of an image is achieved by down-sampling (or sub-sampling):

$$s \downarrow(I)(i, j) = I(si, sj) \quad \text{or} \quad \downarrow s$$

(create a new image by taking every s^{th} pixel of the original image).

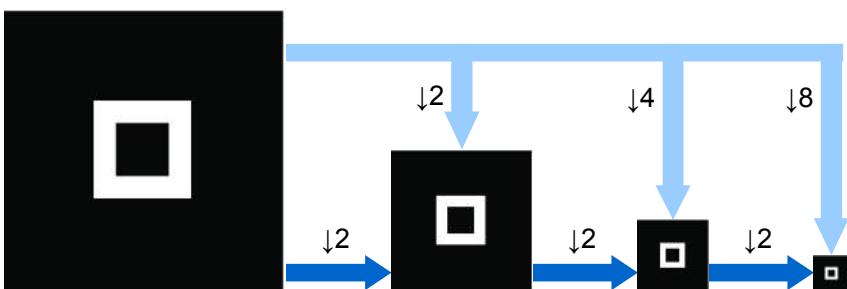
e.g.:



Computer Vision / Low-Level Vision (Artificial)

Image Pyramid creation

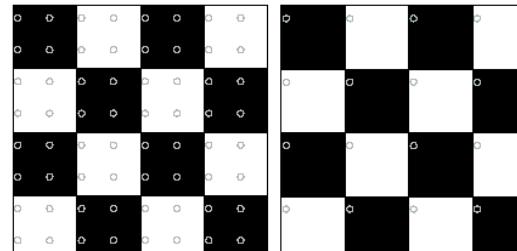
To find image features across a range of scales, we need a number of images at different scales.



Rather than resampling the original image, can sub-sample the previous sub-sampled image (this allows us to write a recursive code).

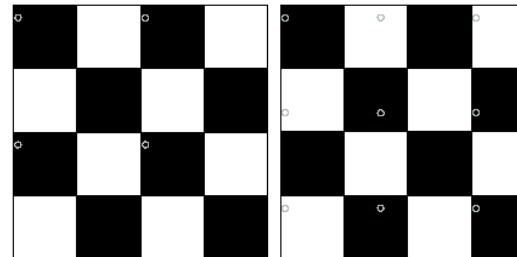
Computer Vision / Low-Level Vision (Artificial)

Down-sampling problem: aliasing



Sample chess board image at each circle:

← Good sampling (output image resembles input)

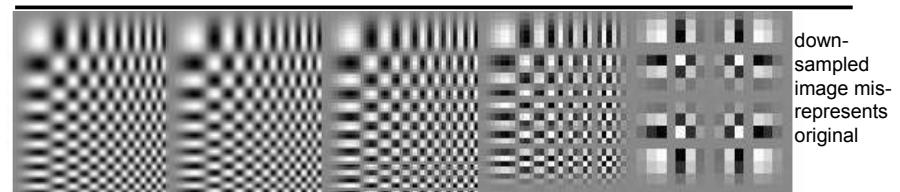


← Bad sampling (aliased)

Sample may not be representative of image region it come from.
Need to take account of larger region then just taking value from a single pixel

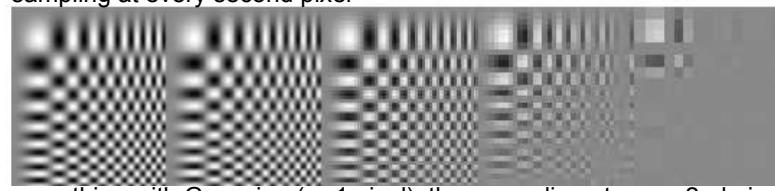
Computer Vision / Low-Level Vision (Artificial)

Down-sampling solution: smoothing

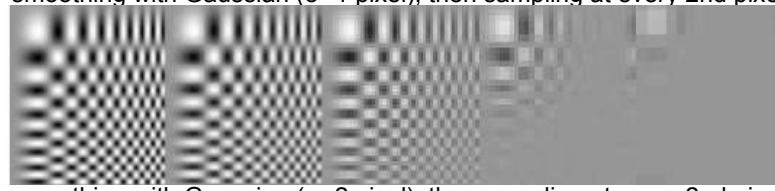


down-sampled
image mis-represents
original

sampling at every second pixel



smoothing with Gaussian ($\sigma=1$ pixel), then sampling at every 2nd pixel



smoothing with Gaussian ($\sigma=2$ pixel), then sampling at every 2nd pixel

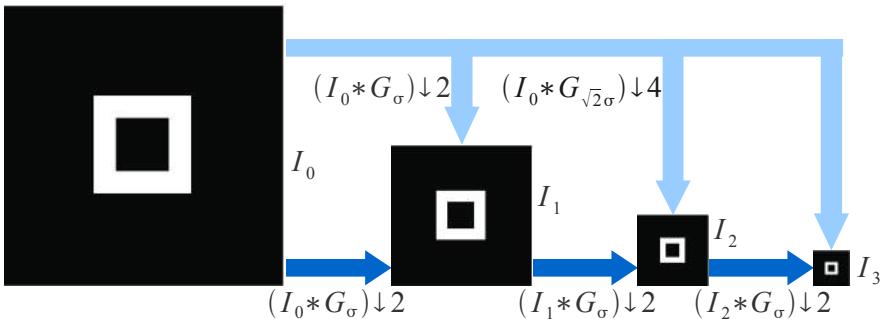
Computer Vision / Low-Level Vision (Artificial)

Gaussian Pyramid creation

If we smooth an image with a Gaussian having a standard deviation of σ , and then convolve the result with a Gaussian having a standard deviation of σ , we get the same result as smoothing the original image with a Gaussian with a standard deviation of $\sqrt{2}\sigma$

i.e. Gaussian * Gaussian=another Gaussian

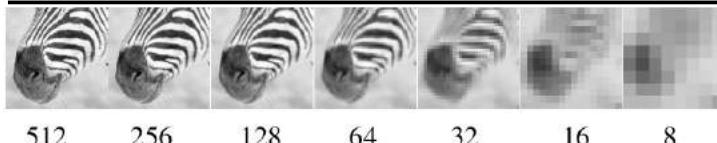
$$G_\sigma * G_\sigma = G_{\sqrt{2}\sigma}$$



$$\text{In general: } I_i = s \downarrow (I_{i-1} * G_\sigma)$$

Computer Vision / Low-Level Vision (Artificial)

Gaussian Pyramid example



Laplacian Pyramid

Recall that:

- a Laplacian of Gaussian (LoG) mask detects intensity discontinuities (e.g. edges) at all orientations
- can be approximated by a Difference of Gaussians (DoG)

A Laplacian image pyramid is an image pyramid that highlights intensity discontinuities at multiple scales.

A Gaussian pyramid already provides us with images that have been convolved with Gaussians of different scales.

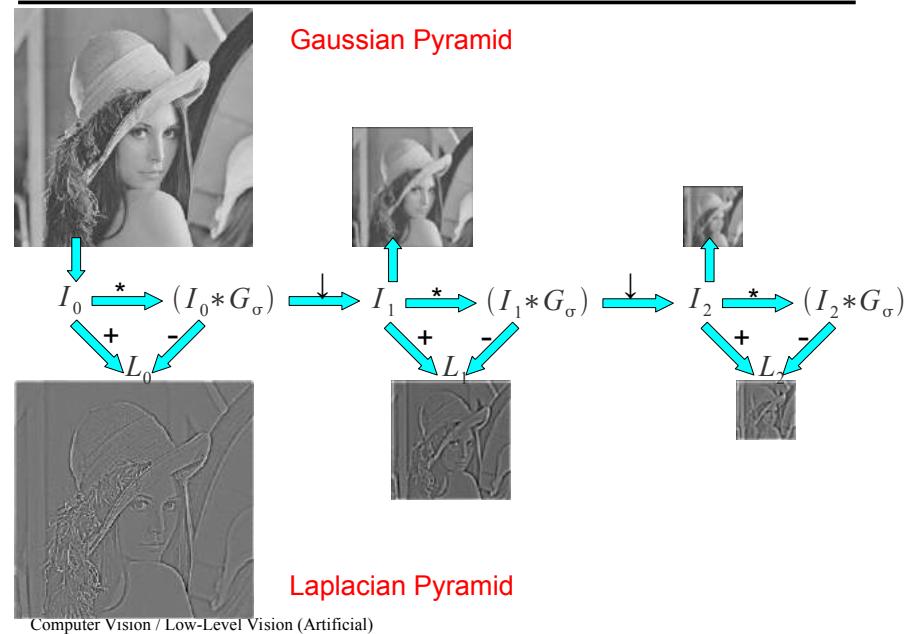
A difference of Gaussians can therefore be obtained by subtracting images taken from the Gaussian pyramid.

$$\text{In general: } L_i = I_i - (I_i * G_\sigma) \quad (\text{an image in the Laplacian pyramid})$$

$$\text{where: } I_i = s \downarrow (I_{i-1} * G_\sigma) \quad (\text{an image in the Gaussian pyramid})$$

Computer Vision / Low-Level Vision (Artificial)

Laplacian Pyramid creation



Computer Vision / Low-Level Vision (Artificial)

Summary

Linear Filtering:

Creates a new image with pixel values that are weighted sums of original pixel values

Requires:

- Convolution – the procedure for calculating the new pixel values
- Mask – the weights used to calculate the new pixel values

Simple (common) Masks:

Smoothing – box, Gaussian

Difference – 1st and 2nd derivatives in x and y directions,

2nd derivative in all directions (Laplacian)

Edge Detection:

Combines smoothing and differencing (to find intensity changes at a certain scale)

- Gaussian derivative mask – 1st derivative of Gaussian in x and y directions
- LoG/DoG mask – 2nd derivative of Gaussian in all directions

Image Pyramids:

Important for describing and searching an image at all scales

- Gaussian pyramid – smoothed and downsampled
- Laplacian pyramid – DoG at each scale

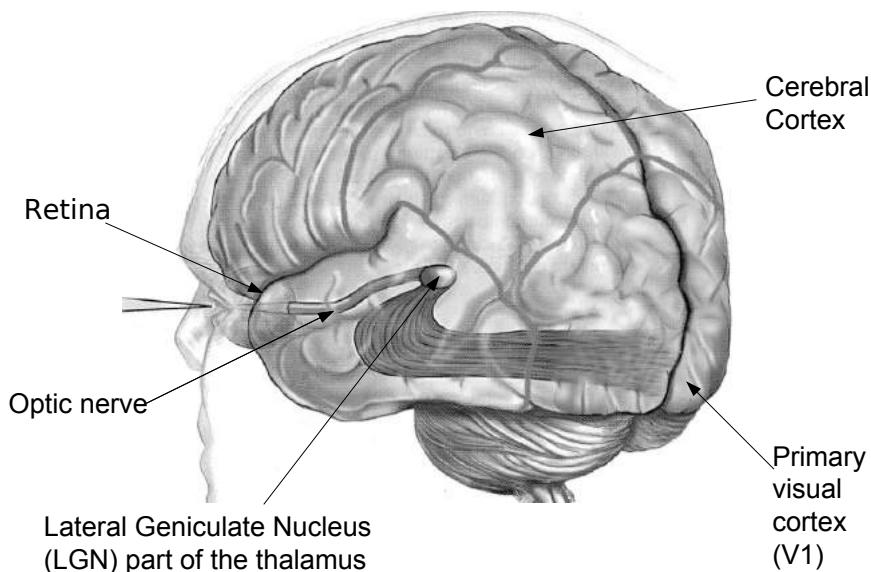
Computer Vision / Low-Level Vision (Artificial)

This Week

- Biological Visual System basics
- Primary visual cortex (V1)
 - physiology
 - input selectivities / classical receptive fields
 - models of orientation selectivity
 - Gabor filters
 - extra-classical receptive field properties
 - contextual influences / change detection
- Mid-level Vision
 - grouping and segmentation
 - top-down and bottom-up influences (Gestalt Laws)
 - extra-classical receptive field properties
 - neural implementation of border ownership and grouping

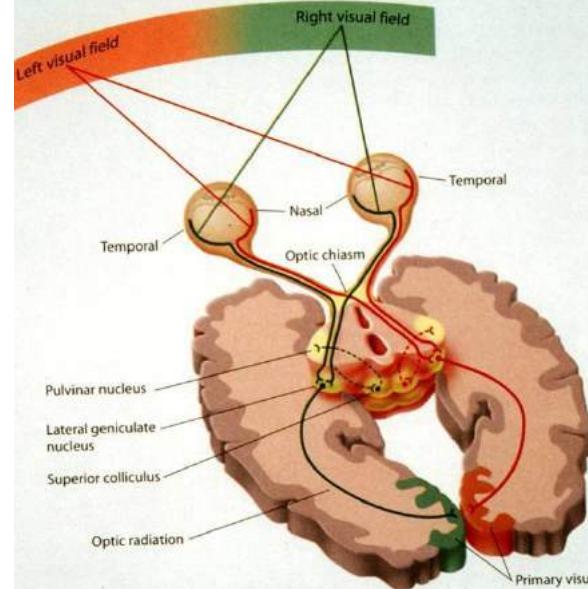
Computer Vision / Low- & Mid-Level Vision / Biological

Human visual system (beyond the retina)



Computer Vision / Low- & Mid-Level Vision / Biological

Pathways from Retina to Cortex



Computer Vision / Low- & Mid-Level Vision / Biological

- The right visual field (RVF) projects to left sides of each retina
- ganglion cells from the left side of the left eye project to the left LGN
- ganglion cells from the left side of the right eye cross over at the optic chiasm and go to the left LGN
- Hence, the RVF projects to the left LGN
- The left LGN projects to the left V1 (striate cortex)
- NOTE: It is not the case that the right eye goes to the left V1, it is the RVF (as seen by both eyes).

What does LGN do?

Lateral geniculate nucleus transmits information from retina to cortex.

- Traditionally viewed as merely a relay station.
- Current evidence suggests it does more than just relay information.
- However, what computation occurs in the LGN is not currently known.

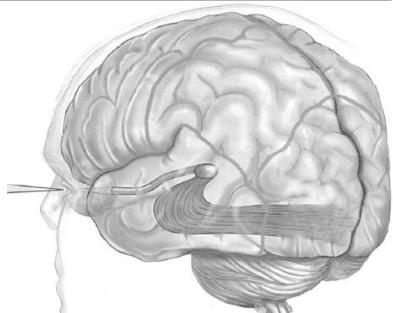
LGN cells have centre-surround RFs, like retinal ganglion cells.

Computer Vision / Low- & Mid-Level Vision / Biological

What does the Cerebral Cortex do?

All higher cognitive functions

- Perception
- Knowledge
- Language
- Memory
- Reasoning
- Decision Making

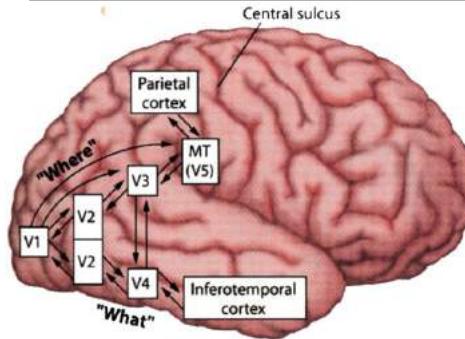


Basic facts:

- A folded sheet approx. 1.7mm thick, with an area of approx. 0.25m²
- Contains 10^{10} neurons (= approx. 10^5 neurons/mm³) and 4×10^{13} synapses
- Is about 2% of human body mass but accounts for 20% of human energy consumption
- Approx. 50% of cerebral cortex is devoted to vision
- Is divided into areas which specialise in different functions. Approx. 30 areas are devoted to different aspects of visual information processing.

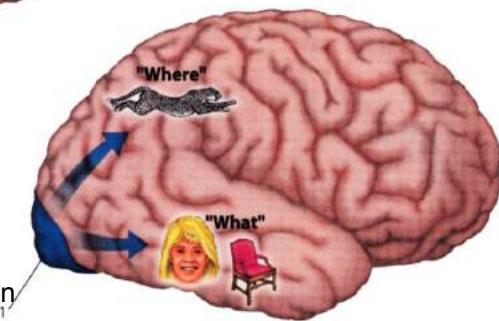
Computer Vision / Low- & Mid-Level Vision / Biological

The Cortical Visual System: pathways



"What" and "Where" pathways
Hierarchically organised:

- simple, local, RFs at V1
- complex, large, RFs in higher areas



Where (or How):

- V1 to parietal cortex
- spatial / motion information

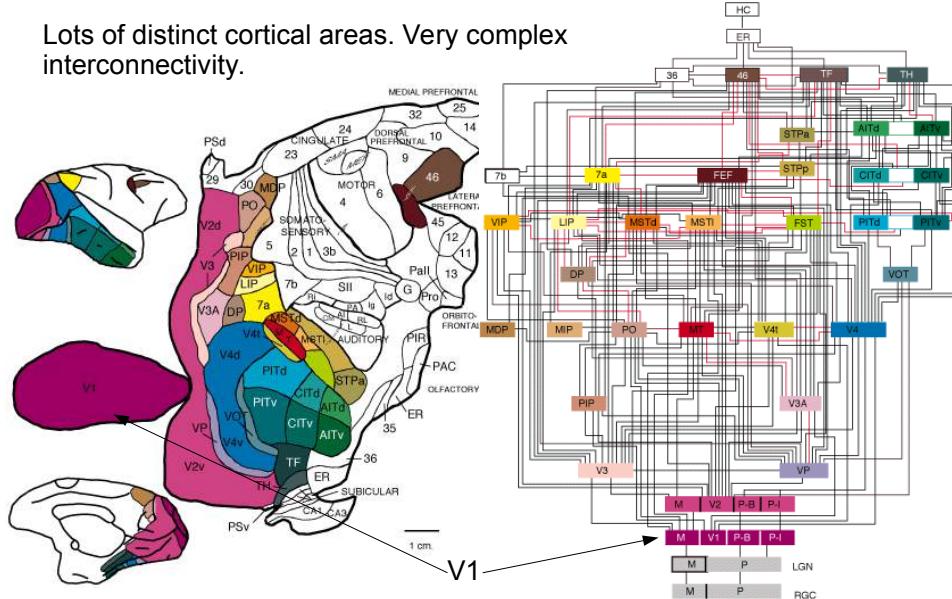
What

- V1 to inferotemporal cortex
- identity / category information

Computer Vision / Low- & Mid-Level Vision / Biological

The Cortical Visual System: areas

Lots of distinct cortical areas. Very complex interconnectivity.



Computer Vision / Low- & Mid-Level Vision / Biological

Content

- Primary visual cortex (V1)

- physiology
 - input selectivities / classical receptive fields
- organisation
 - hypercolumns

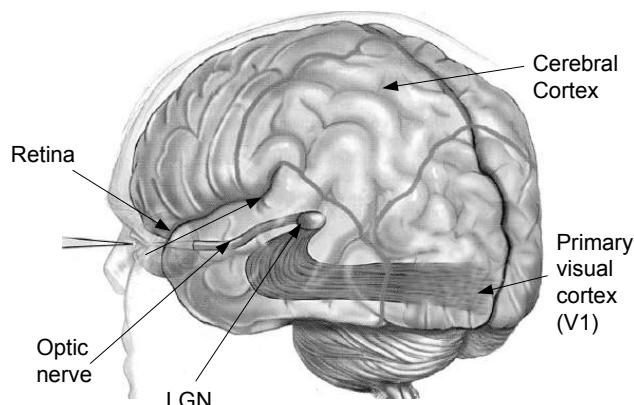
Computer Vision / Low- & Mid-Level Vision / Biological

Cortical area V1

LGN neurons mainly project to the primary visual cortex

Primary visual cortex is also known as:

- V1
- striate cortex
- area 17



V1 therefore performs the initial (low-level) processing on the incoming information.

Processing is carried out by cells with a larger range of receptive field properties than are found in the retina and LGN.

Computer Vision / Low- & Mid-Level Vision / Biological

V1 RFs: range of selectivities

V1 cells have receptive fields selective for:

- colour
- orientation
- direction of motion
- spatial frequency
- eye of origin
- binocular disparity
- position

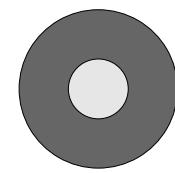
Some of these properties are similar to those of cells in the LGN and retinal ganglion cells (e.g., colour, eye of origin, position).

Other properties are seen for the first time in V1 (e.g., orientation selectivity and binocular disparity, direction of motion).

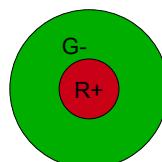
Computer Vision / Low- & Mid-Level Vision / Biological

V1 RFs: centre-surround / colour

Centre-surround / colour opponent RFs are similar to those in the LGN/retina and perform the same function: detecting changes in intensity or regions of uniform colour.



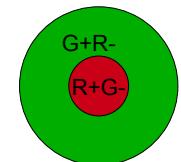
Broad band centre-surround



Colour opponent centre-surround



Colour opponent centre only



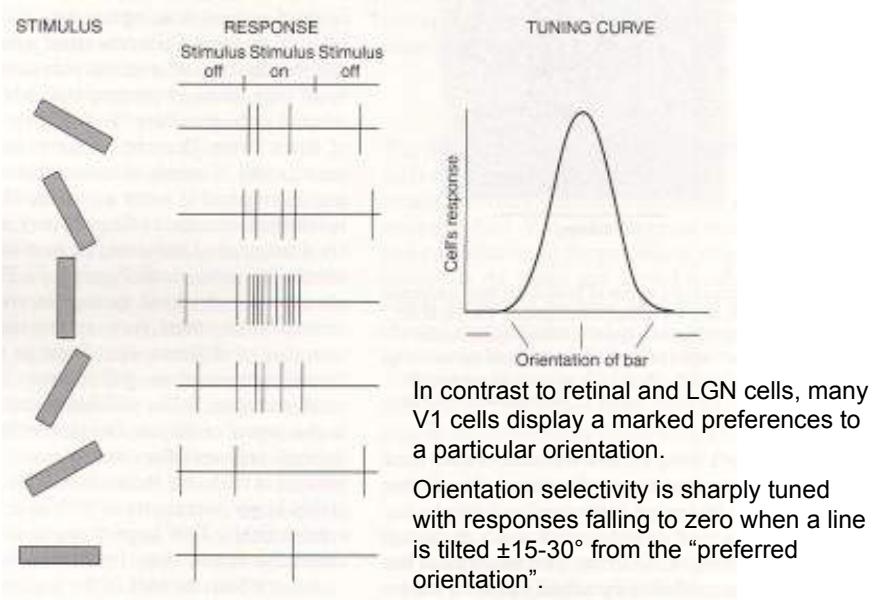
Double-opponent

Double-opponent (DO) cells are not seen prior to V1. They detect locations where colour changes (e.g. where centre is red and background green).

have larger receptive fields than simple opponent cells

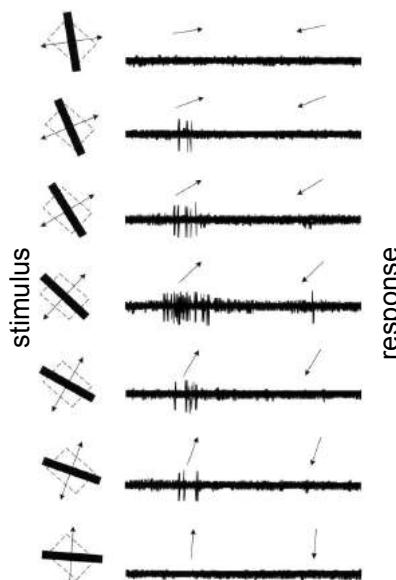
Computer Vision / Low- & Mid-Level Vision / Biological

V1 RFs: orientation selectivity



Computer Vision / Low- & Mid-Level Vision / Biological

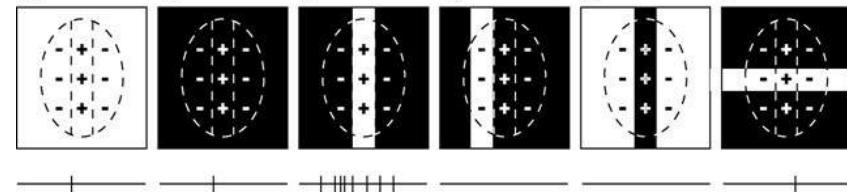
V1 RFs: orientation (simple cells)



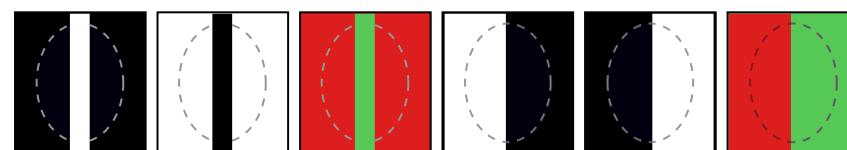
Computer Vision / Low- & Mid-Level Vision / Biological

V1 RFs: orientation (simple cells)

Optimum response to an appropriately oriented stimulus, with correct contrast, placed at a specific position within the receptive field:



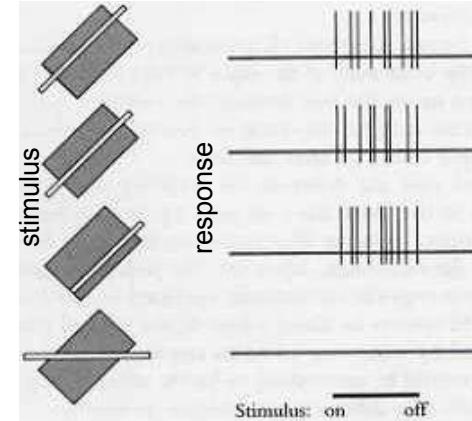
Different cells can be selective for different stimuli:



In general, simple cells act as edge and bar detectors.

Computer Vision / Low- & Mid-Level Vision / Biological

V1 RFs: orientation (complex cells)



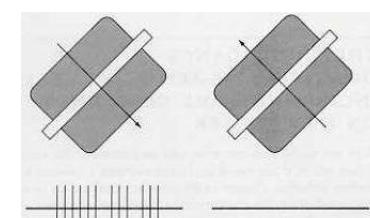
Complex cells act as edge and bar detectors with some tolerance to location.

Computer Vision / Low- & Mid-Level Vision / Biological

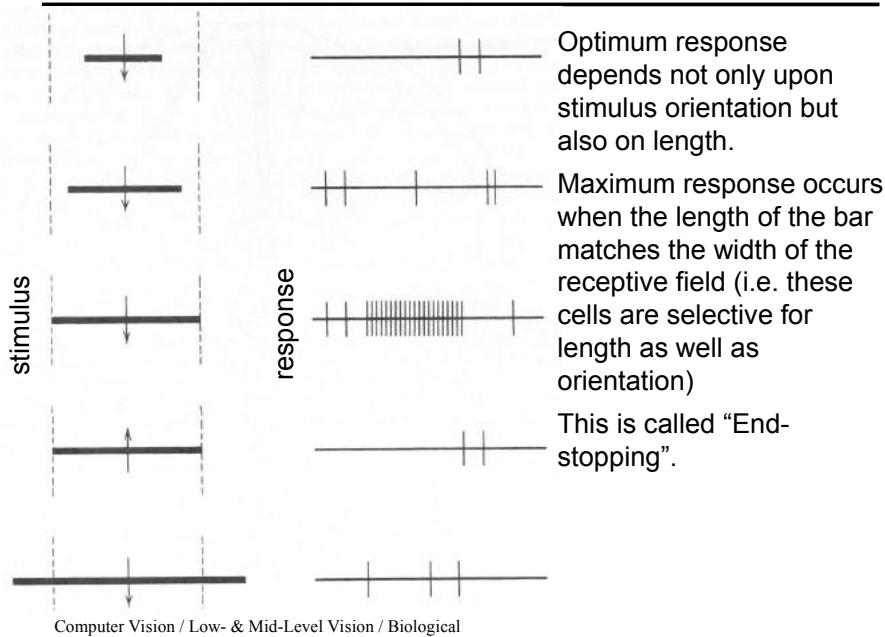
Optimum response to an appropriately oriented stimulus, of any contrast, placed anywhere within the receptive field.

Complex cells thus have a greater positional invariance compared to simple cells.

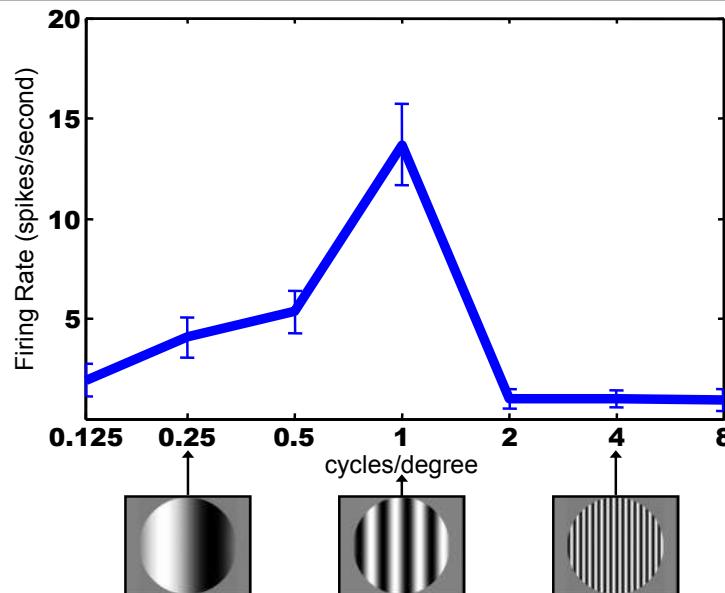
Usually they respond most strongly to moving bars/edges and can be direction selective:



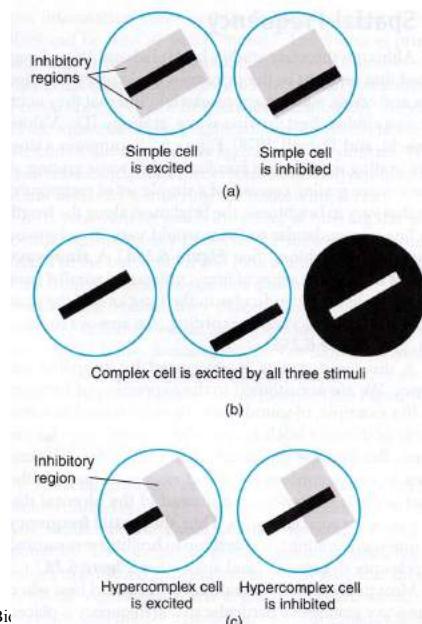
V1 RFs: orientation (hyper-complex cells)



V1 RFs: spatial frequency tuning



V1 RFs: orientation selectivity (summary)



V1 RFs: eye of origin

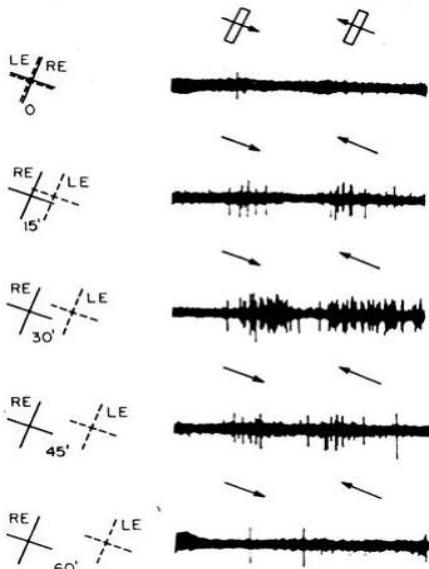
Monocular cells

Retinal and LGN cells receive input from one eye only.
Cells in V1 that receive the input from LGN are also monocular.

Binocular cells

Other cells in V1 are binocular: they receive input (via monocular cells) from the two eyes.
Binocular cells have two receptive fields (Left eye & Right eye) and these are matched in type (e.g. same direction of motion preference, same orientation preference, both simple or both complex).
Hence, binocular cells respond maximally when corresponding regions in each eye are stimulated by stimuli of similar appearance.

V1 RFs: disparity



Computer Vision / Low- & Mid-Level Vision / Biological

For this neuron, maximum response occurs when a bar is presented simultaneously to both eyes, with the correct orientation, direction of motion and in the correct position on each retina.

V1 RFs: conjunctions

V1 cells have receptive fields selective for:

- colour
- orientation
- direction of motion
- spatial frequency
- eye of origin
- binocular disparity
- position

Some cells are responsive to more than one of these properties

e.g. orientation and direction of motion,
orientation and colour

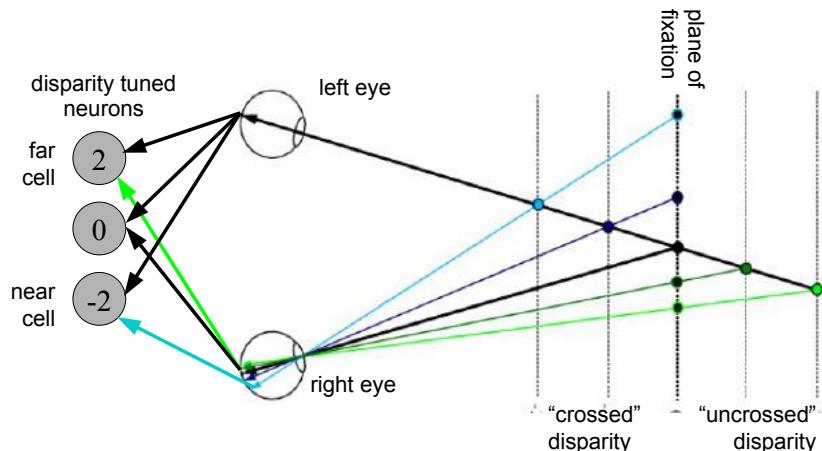
All are selective for at least one of these properties **plus** position (as the stimulus needs to be at a particular location on the retina).

Computer Vision / Low- & Mid-Level Vision / Biological

V1 RFs: disparity

Binocular cells are tuned to disparity: the difference in the preferred location in each eye

This enables them to encode the depth of the stimulus



Computer Vision / Low- & Mid-Level Vision / Biological

V1 organisation: Hypercolumns

All neurons with RFs at the same location on the retina are located in the same region of V1.

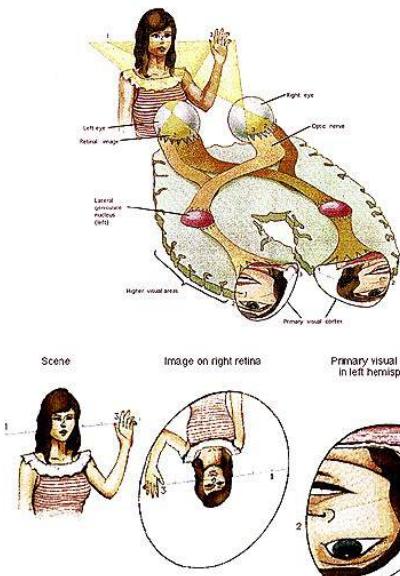
Approx. 1mm^2 of V1 is required to represent the whole range of RF types for one particular position.

Such a 1mm^2 area of V1 is called a "hypercolumn"

Hence, each hypercolumn contains the requisite neural machinery to simultaneously analyse multiple attributes of an image (colour, orientation, direction of motion, spatial frequency, eye of origin, binocular disparity) falling on a localised region of the retina.

Computer Vision / Low- & Mid-Level Vision / Biological

V1 organisation: retinotopic maps



Adjacent hypercolumns analyse information from adjacent areas of retina.

Hence, the spatial position of the ganglion cells within the retina is preserved by the spatial organisation of the neurons within V1.

This spatial layout is called **retinotopic** organization because the topological organization of the receptive fields in V1 parallels the organization of the retina.

The map is distorted primarily due to cortical magnification of central vs. peripheral areas.

There is more cortical area (more hypercolumns) devoted to the fovea than peripheral vision.

This mirrors the fact that the vast majority of retinal ganglion cells are devoted to the fovea.

Content

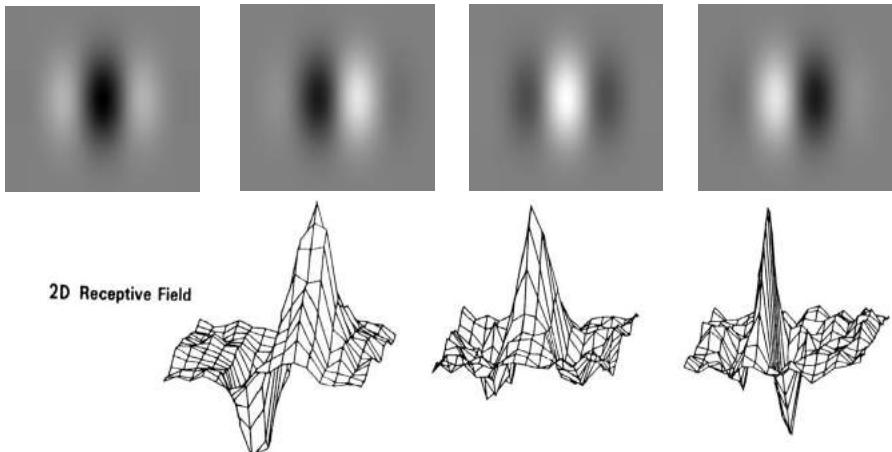
• Gabor filters

- models of V1 orientation selectivity
 - edge-detection
 - image components
 - applications in computer vision

Computer Vision / Low- & Mid-Level Vision / Biological

Gabor: model of orientation selective RFs

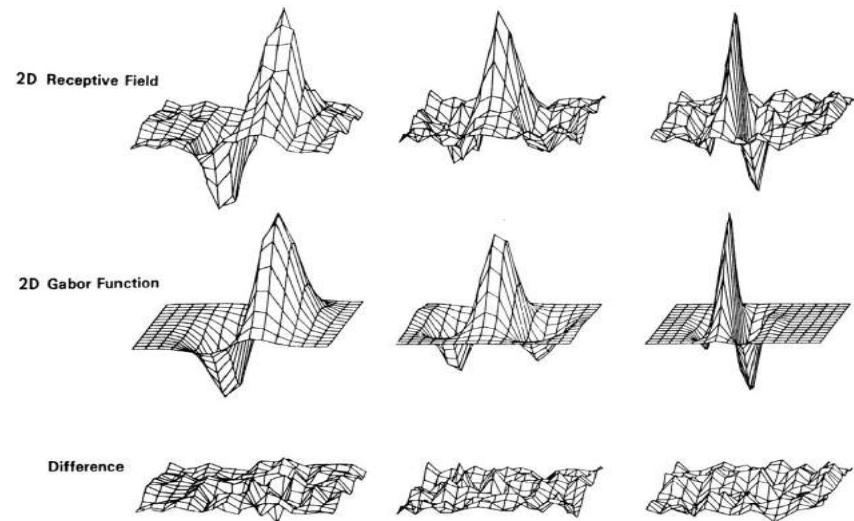
Simple cells RFs actually look more like this:



These are very similar to a mathematical function called a Gabor.

Computer Vision / Low- & Mid-Level Vision / Biological

Gabor: model of orientation selective RFs



Computer Vision / Low- & Mid-Level Vision / Biological

Gabor: model of orientation selective RFs

A 2-D Gabor function is a Gaussian multiplied by a sinusoid.

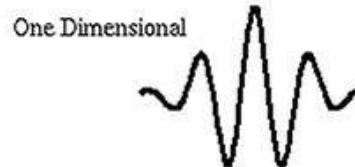
$$G(x, y) = \exp\left(-\frac{x'^2 + y'^2}{2\sigma^2}\right) \cos(2\pi x' f + \psi)$$

where:

$$x' = x \cos \theta + y \sin \theta$$

$$y' = -x \sin \theta + y \cos \theta$$

Gabor Function

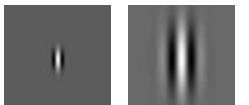


Computer Vision / Low- & Mid-Level Vision / Biological

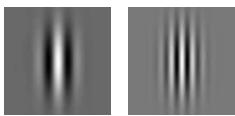
Gabor: model of orientation selective RFs

The Gabor function has 5 parameters:

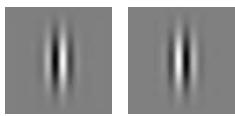
σ = the standard deviation of the Gaussian function



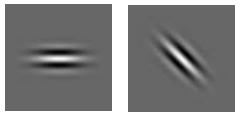
f = the frequency of the sinusoidal function



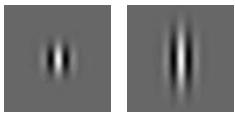
ψ = the phase of the sinusoidal function



θ = the orientation



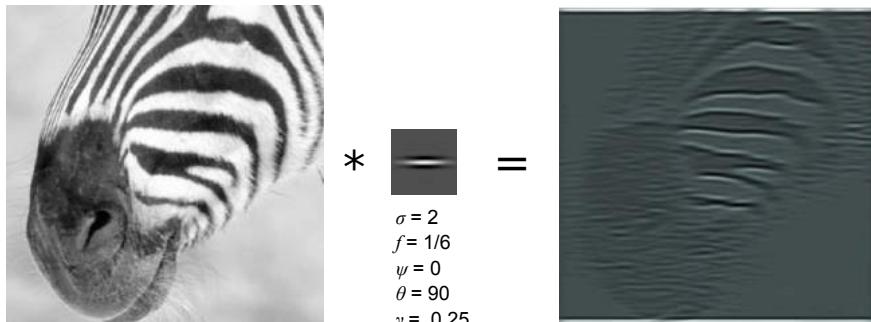
γ = the spatial aspect ratio



Computer Vision / Low- & Mid-Level Vision / Biological

Gabor: model example

To simulate the activity of **simple cells** in response to an image, we can convolve the image using a Gabor function as the mask.

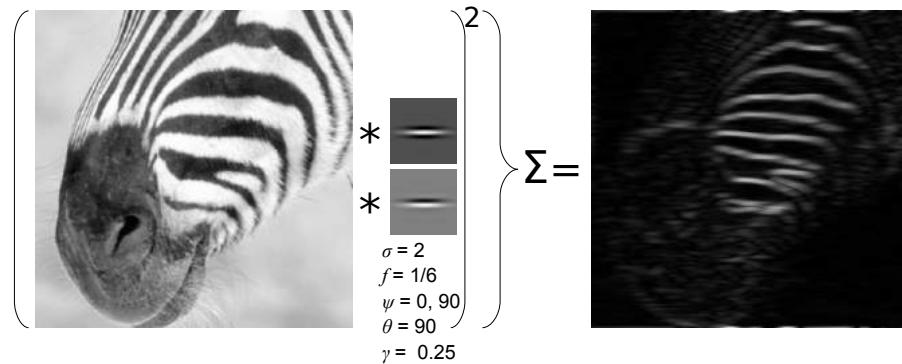


This provides a crude model of the responses of all simple cells selective for the same orientation, spatial frequency and phase across all cortical hypercolumns.

Computer Vision / Low- & Mid-Level Vision / Biological

Gabor: model example

To simulate the activity of **complex cells** in response to an image, we can pool the outputs generated by convolving the image with Gabor functions of varying phases.



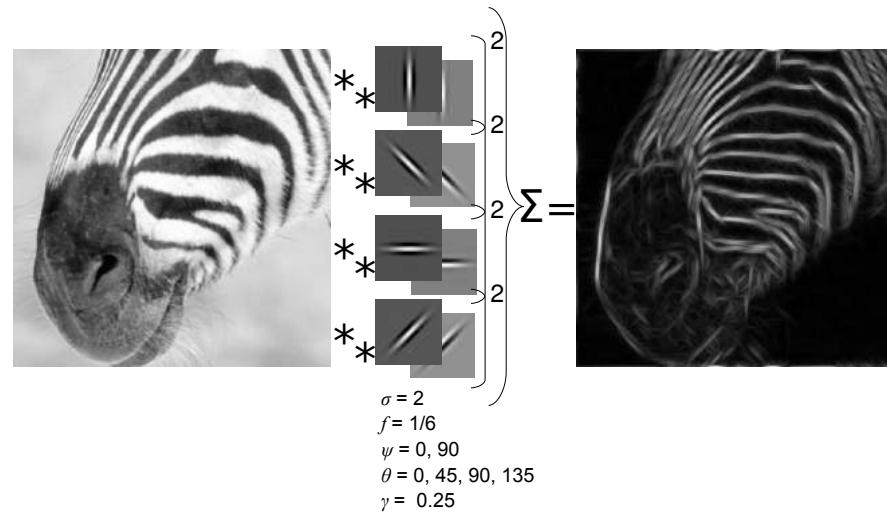
The **energy model** takes the square root of the sum of squared output of a quadrature pair of Gabor filters.

Result is invariant to phase.

Computer Vision / Low- & Mid-Level Vision / Biological

Gabor: model example

To detect edges at multiple orientations, we can sum the outputs from simulated complex cells at multiple orientations.



Computer Vision / Low- & Mid-Level Vision / Biological

Multiscale Gabors: wavelet transforms

To detect edges at multiple spatial scales, we can use Gabor functions with different frequencies and standard deviations.

Convolving a signal (e.g. an image) with a family of similar masks sensitive to different frequencies is known as a “Continuous Wavelet Transform”.

The “wavelet” is the mathematical function used to generate the masks (e.g. a Gabor).

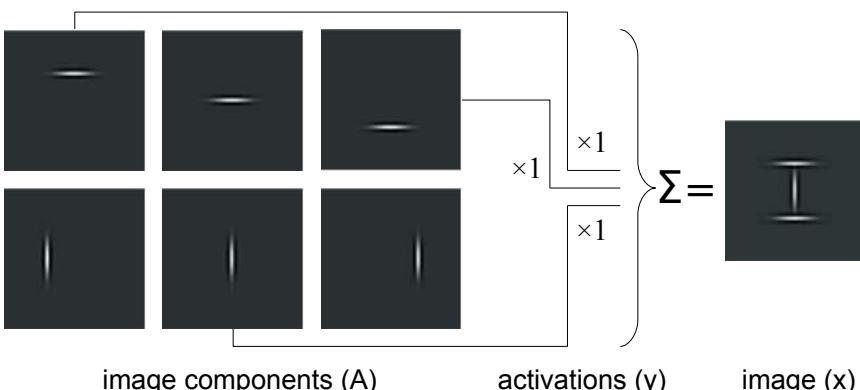
Hence, we can think of the simple cells in V1 as performing a continuous wavelet transform.

Computer Vision / Low- & Mid-Level Vision / Biological

Gabors: as image components

We can think of an image being made up of a superposition of various image components or elementary features.

e.g. A letter I as a combination of three lines (Gabor image components):

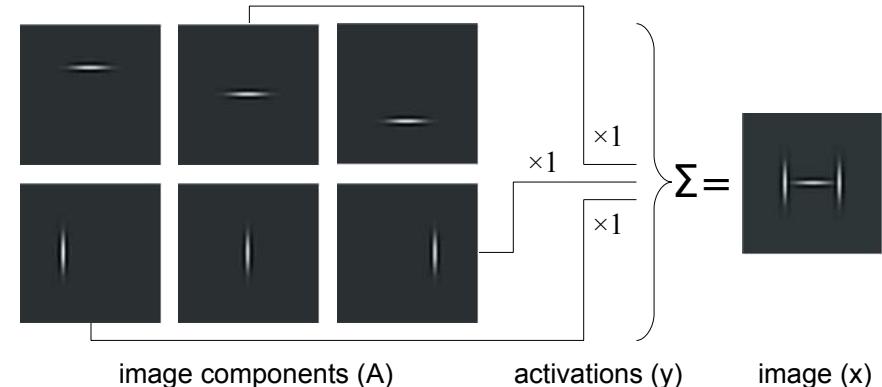


Computer Vision / Low- & Mid-Level Vision / Biological

Gabors: as image components

Combining different components in different proportions can generate numerous different images.

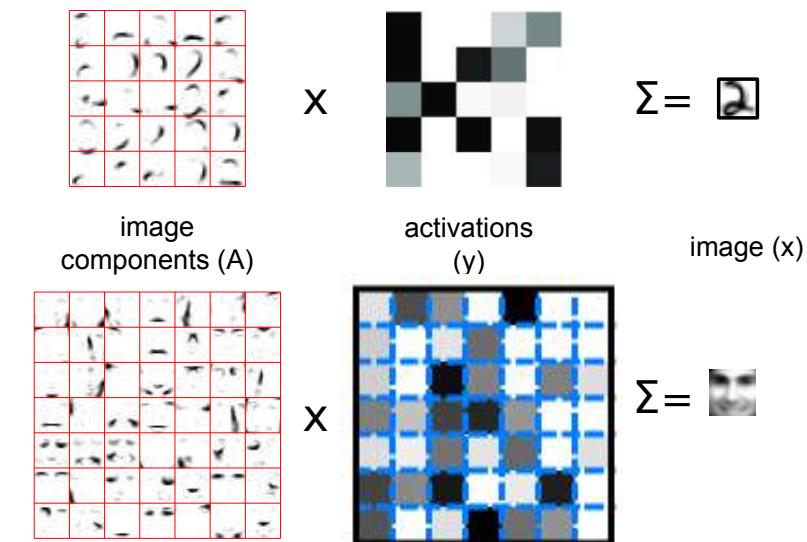
e.g. A letter H as a combination of three lines (Gabor image components):



Computer Vision / Low- & Mid-Level Vision / Biological

Image components

This is a very general concept, e.g.:



Computer Vision / Low- & Mid-Level Vision / Biological

Image components

Converting arrays to vectors, e.g.

-1	1	0.5
0	-2	-1
0.5	-1	-1
-1	1	0

→

-1
0
0.5
-1
1
-2
-1
1
0.5
-1
-1
0

allows this idea to be expressed mathematically, as:

$$A y \approx x$$

where:

A = is an m by n matrix of weight values the columns of which represent image components

y = is a n by 1 vector describing the activation of each component in the image

x = is a m by 1 vector representing the image

Computer Vision / Low- & Mid-Level Vision / Biological

Image components

If we have lots of images, then this equation becomes:

$$A Y \approx X$$

where:

A = is an m by n matrix of weight values the columns of which represent image components

Y = is a n by p matrix each column of which contains the activation of each component in the corresponding image

X = is a m by p matrix each column of which contains the pixel values for one image

Computer Vision / Low- & Mid-Level Vision / Biological

Image components

Generally we know X , and want to determine how this set of images can be represented, i.e. we want to find the factors A and Y of X .

$$A Y \approx X$$

Finding the factors A and Y is an ill-posed problem (there are lots of possible solutions).

By placing additional constraints on the factors A and Y different solutions, with different properties, can be found.

There are several standard “Matrix factorisation” methods that can find A and Y under different constraints. e.g.:

- Principal Component Analysis (PCA)
- Independent Component Analysis (ICA)
- Non-negative Matrix Factorization (NMF)

Computer Vision / Low- & Mid-Level Vision / Biological

Gabors as components of natural images

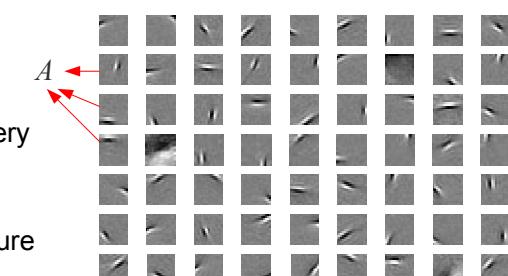
What are the components of natural images?

If we create X by randomly selecting small image patches



Then find the factors using two constraints:

1. That information be preserved (i.e. that $\|X - AY\|$ is minimised)
2. The representation be sparse (i.e. the number of non-zero values in each column of Y is minimised)



Then the columns of A are very similar to Gabor functions.

Gabors therefore appear to capture the intrinsic structure of natural images

Computer Vision / Low- & Mid-Level Vision / Biological

Gabors as efficient code

The sparsity constraint used to find the components of natural images, means that only a few components are present in each image.

If components are represented by neurons, then this means that only a few neurons need to be active in order to represent an image.

Hence, Gabor RFs result in an efficient code which minimises the number of active neurons needed to transmit a given signal.

Computer Vision / Low- & Mid-Level Vision / Biological

Image components: Computer Vision Applications

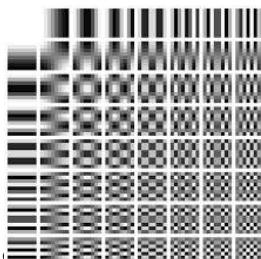
Image Compression

The ability to reconstruct an image using image components allows an image to be efficiently encoded.

e.g. jpeg

Reconstructs each image patch as a linear combination of a set of image components

image components
are cosine functions
not Gabor functions



$A y \approx x$
activations are
quantized rather
than sparse

y requires less storage
space than x

Computer Vision / Biological

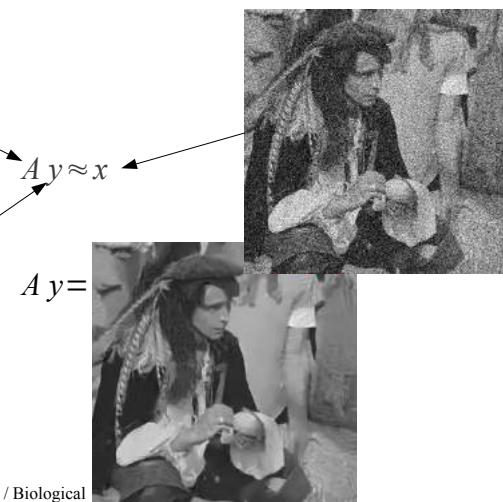
Image components: Computer Vision Applications

Image Denoising

Reconstruct a noisy image using image components produces a reconstructed image that has less noise.

image components
are Gabor functions
(or curvelets,
wedgelets, or other
functions)

activations are
found that produce
the most accurate
reconstruction of
the noisy image
(i.e. that minimise
 $\|x-Ay\|$)



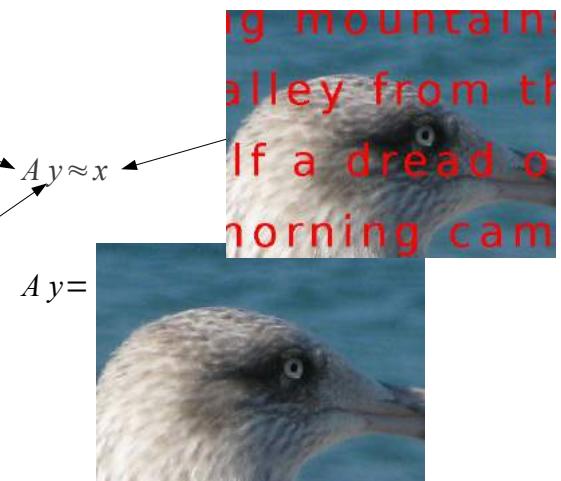
Computer Vision / Low- & Mid-Level Vision / Biological

Image components: Computer Vision Applications

Image Inpainting

Reconstruct missing parts of an image using a sparse-subset of image components that represent non-corrupted image patches

image components
learnt from non-
corrupted parts of
image.



Computer Vision / Low- & Mid-Level Vision / Biological

Content

• Non-Classical Receptive Fields in V1

- neural implementation
- influences on perception

Computer Vision / Low- & Mid-Level Vision / Biological

V1 non-classical RFs

Classical Receptive Field (cRF) = the region of visual space that can elicit a response from a neuron.

However, this response can be modulated by stimuli appearing outside the neurons classical receptive field.

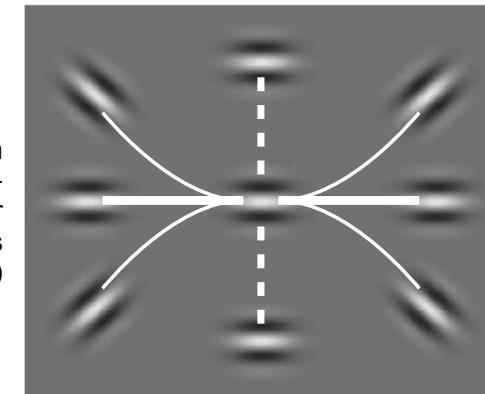
The region of visual space that can modulate the response is called the non-classical receptive field (ncRF).

The result is that neuronal responses are influenced by **context**, i.e., neuronal activity at one location depends on activity at distant locations.

Computer Vision / Low- & Mid-Level Vision / Biological

V1 non-classical RFs (orientation tuned cells)

ncRF caused by lateral (and feedback) connections

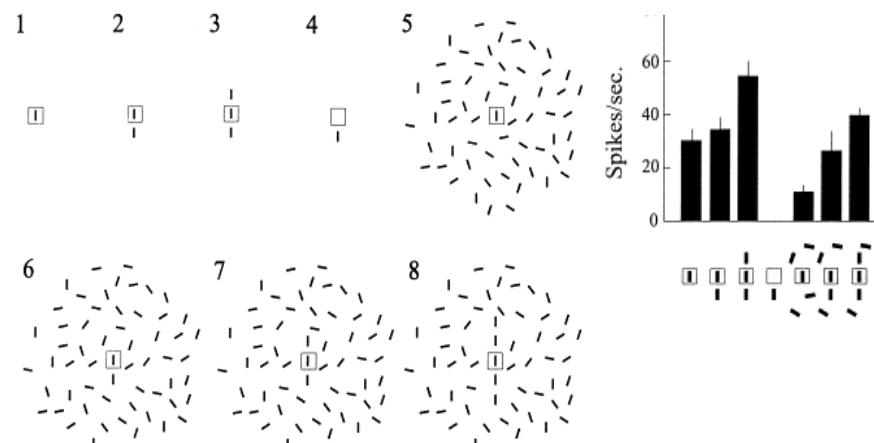


Thick lines indicate strongest connections (greatest influence)

This pattern of lateral connectivity is sometimes called the “association field”

Computer Vision / Low- & Mid-Level Vision / Biological

V1: contextual influences



- context has no effect in isolation
- context can enhance response
- context can reduce response

Computer Vision / Low- & Mid-Level Vision / Biological

V1: contextual influences

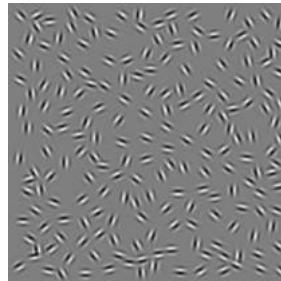
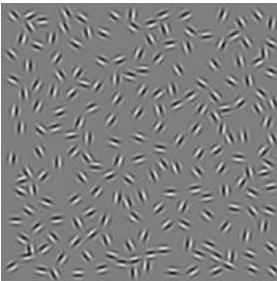
Collinear Facilitation

Which of the central Gabor patches is easier to see?



Contour Integration

Which of these images contains a straight line?

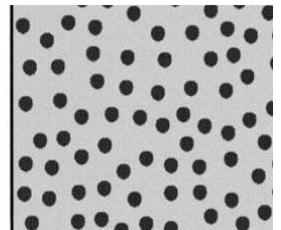
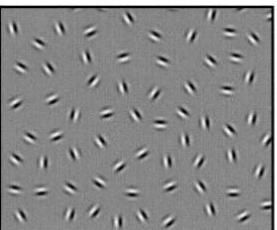
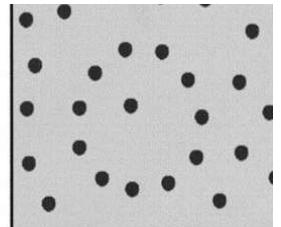
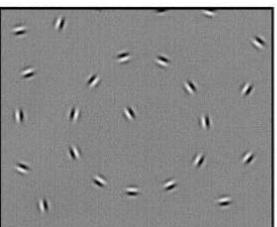


Computer Vision / Low- & Mid-Level Vision / Biological

V1: contextual influences

Contour Integration

Which of these images contains a circle?



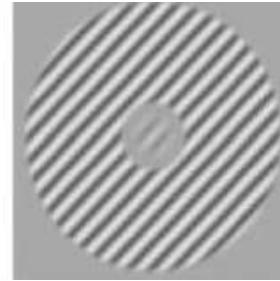
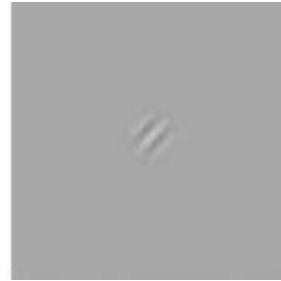
Lateral excitatory connections enhance responses to aligned elements, making these easier to perceive.

Computer Vision / Low- & Mid-Level Vision / Biological

V1: contextual influences

Surround Suppression

Which of the central Gabor patches is easier to see?

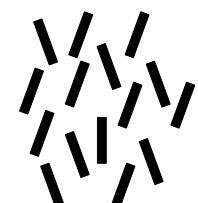
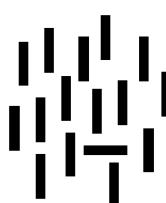


Computer Vision / Low- & Mid-Level Vision / Biological

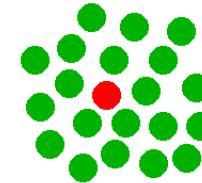
V1: contextual influences

Pop-out

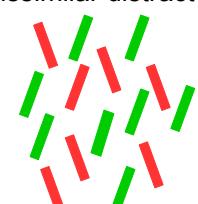
Which is the odd element in each figure?



dissimilar distractors



feature search



conjunction search

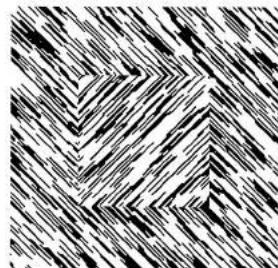
Lateral inhibitory connections suppress responses to similar elements, making dissimilar elements easier to perceive.

Computer Vision / Low- & Mid-Level Vision / Biological

V1: contextual influences

Texture Segmentation

Where is there a boundary in each figure?

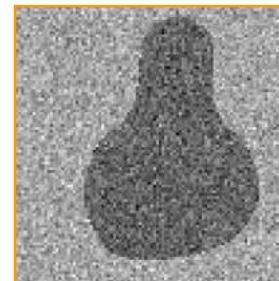


Similar to pop-out, but with two regions. Lateral inhibitory connections suppress responses to similar elements. At the boundary between dissimilar elements neurons receive less inhibition making it easier to perceive.

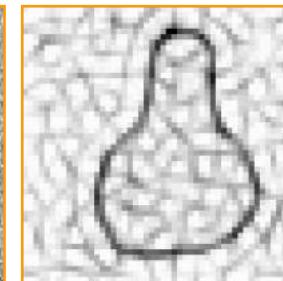
Modelling contextual influences in V1

Contour Detection

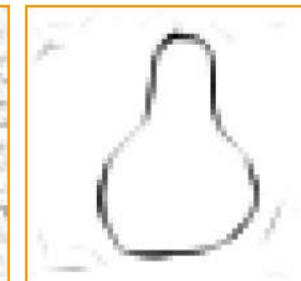
Input image



filtering



filtering + lateral



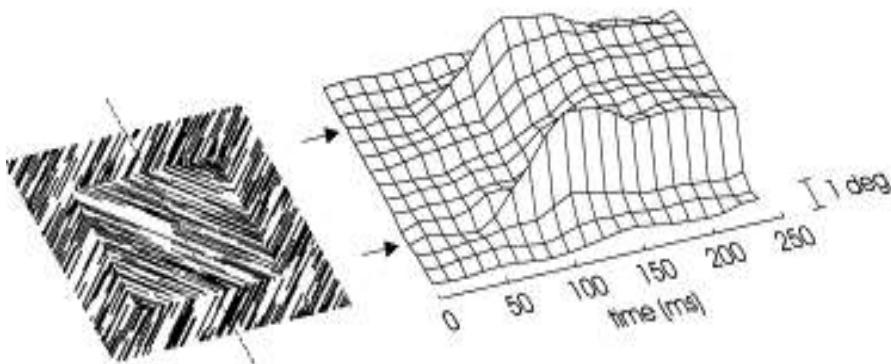
Edge-detection methods based on convolution fail to detect only meaningful edges (such as object boundaries).

More complex operations, like those due to the ncRF, are required for accurate identification of meaningful edges.

Computer Vision / Low- & Mid-Level Vision / Biological

V1: contextual influences

The neural correlates of this effect:



Lateral inhibitory connections suppress responses to similar elements, making boundary between dissimilar elements easier to perceive.

Computer Vision / Low- & Mid-Level Vision / Biological

Computer Vision / Low- & Mid-Level Vision / Biological

Content

• Mid-level Vision

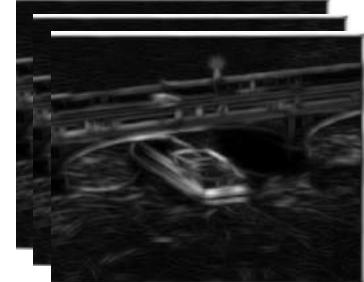
- grouping and segmentation

Mid-Level Vision

Image Formation



Low-Level Vision



noise suppression
edge enhancement
feature extraction
efficient coding



A boat passing under
Westminster Bridge

object recognition
object classification
object localisation
scene understanding

High-Level Vision

Computer Vision / Low- & Mid-Level Vision / Biological

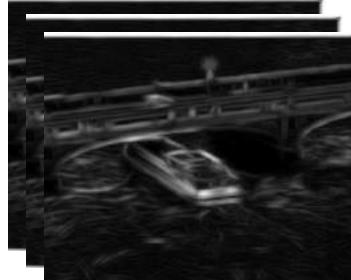
Computer Vision / Low- & Mid-Level Vision / Biological

Mid-Level Vision

Image Formation



Low-Level Vision



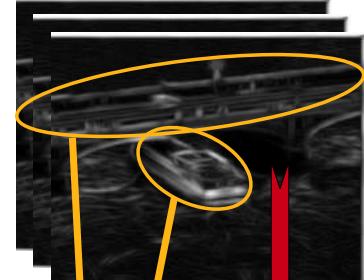
noise suppression
edge enhancement
feature extraction
efficient coding

Mid-Level Vision

Image Formation



Low-Level Vision



noise suppression
edge enhancement
feature extraction
efficient coding

A boat passing under
Westminster Bridge

object recognition
object classification
object localisation
scene understanding

Mid-Level Vision

Computer Vision / Low- & Mid-Level Vision / Biological

Computer Vision / Low- & Mid-Level Vision / Biological

High-Level Vision

Mid-Level Vision

The role of Mid-Level vision is to:

- **group** together those elements* of an image that “belong together”, and to
- **segment** (differentiate) these elements* from all others.

*elements (or tokens) can be whatever we need to group (pixel intensities, pixel colours, edges, features, etc.)

Grouping and segmentation are “two sides of the same coin”: doing one implies doing the other

Terms are used interchangeably, and there are also a number of other names of the same process...

Grouping and Segmentation: Names

- Image Parsing
- Perceptual Organisation
- Binding
- Perceptual Grouping
- Figure-Ground segmentation (restricted case):
 - segment one object (the figure) from everything else (the ground / background)

Content

- **Gestalt Laws**

- image segmentation in the brain
- top-down influences
- bottom-up influences (the Gestalt Laws)
 - neural implementation via V1 lateral connections

Computer Vision / Low- & Mid-Level Vision / Biological

Grouping and Segmentation: Approaches

Top-down (coming from internal knowledge, prior experience)

- elements belong together because they lie on the same object

Bottom-up (coming from the image properties)

- elements belong together because they are similar or locally coherent

These approaches are NOT mutually exclusive

Both bottom-up and top-down influences are required for general purpose image segmentation

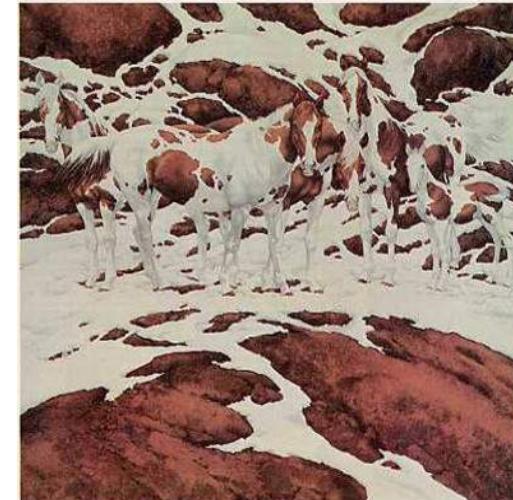
In humans we can observe (with the right choice of image) the influences of individual bottom-up and top-down cues to perceptual grouping.

Computer Vision / Low- & Mid-Level Vision / Biological

Top-down Influences: Knowledge

Meaningfulness or Familiarity:

Elements that, when grouped, form a familiar object tend to be grouped together.



Computer Vision / Low- & Mid-Level Vision / Biological

Top-down Influences: Expectation

Motion Pareidolia

Nicolas Davidenko, Yeram Cheong, Jake Smith
UC Santa Cruz

Computer V

Bottom-up Influences: Gestalt Laws

Many bottom-up factors influence how elements are grouped.

These influences are called the “Gestalt Laws” or the “Gestalt Grouping Principles”

Gestalt Laws are not really laws (that must be obeyed) but heuristics (“rules of thumb” that are often obeyed)

Original Gestalt Laws (proposed in early 20th century) include:

- Proximity
- Similarity
- Closure
- Continuity (or Good Continuation)
- Common Fate
- Symmetry

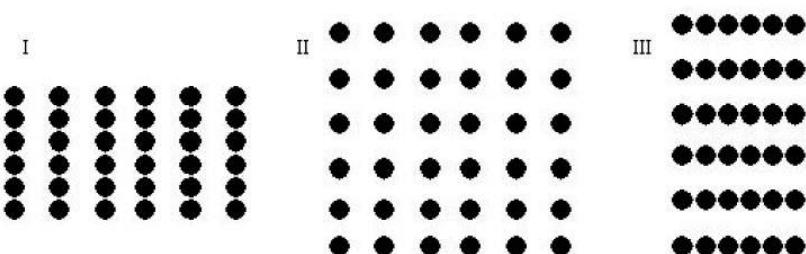
More Recent Laws (hence, not *true* Gestalt Laws) include:

- Common Region
- Connectivity

Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Proximity

Elements that are near to each other tend to be grouped.

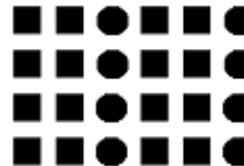


Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Similarity

Elements that are similar tend to be grouped together.

Similar shape



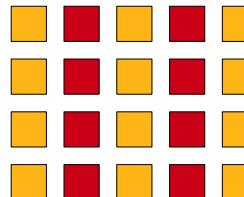
Similar orientation



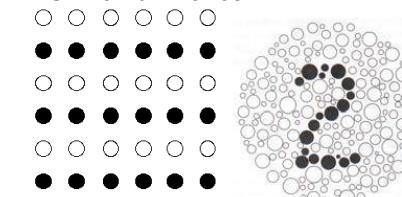
Similar size



Similar colour



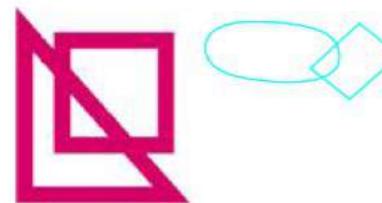
Similar luminance



Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Closure

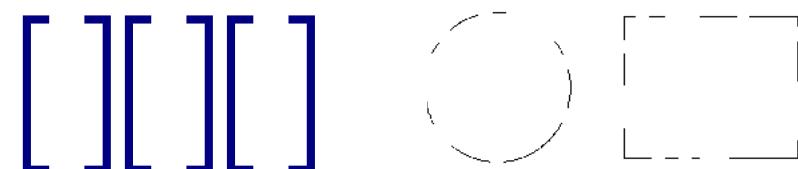
Elements that, when grouped, result in closed boundaries tend to be seen as belonging together.



These are perceived as a square and triangle (left), and an ellipse and a rectangle (right), not as a combination of strange shapes, e.g.:



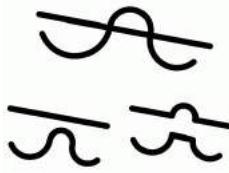
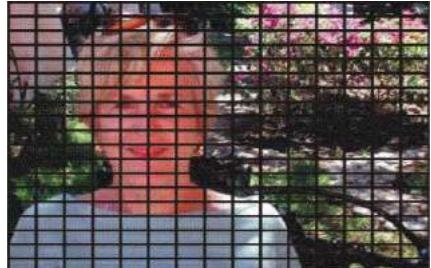
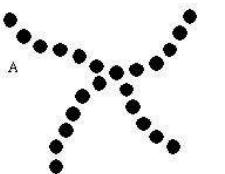
Potential closure is sufficient to result in grouping:



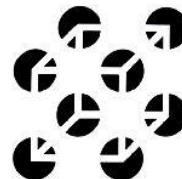
Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Continuity

Elements that, when grouped, result in straight or smoothly curving lines, or smooth surfaces, tend to be seen as belonging together



We see this...but not this
Continuity

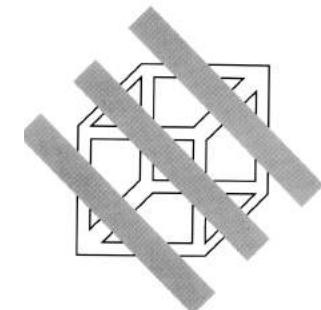
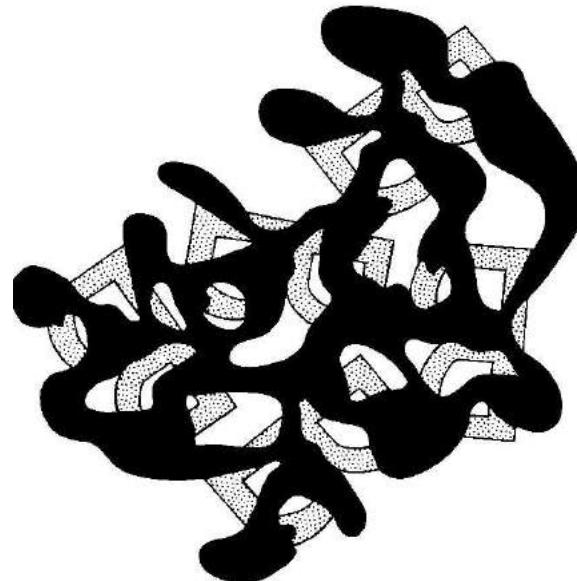


We even “invent” illusory contours (groups of white elements) to conform to our expectations about continuity.

Computer Vision / Low- & Mid-Level Vision / Biological

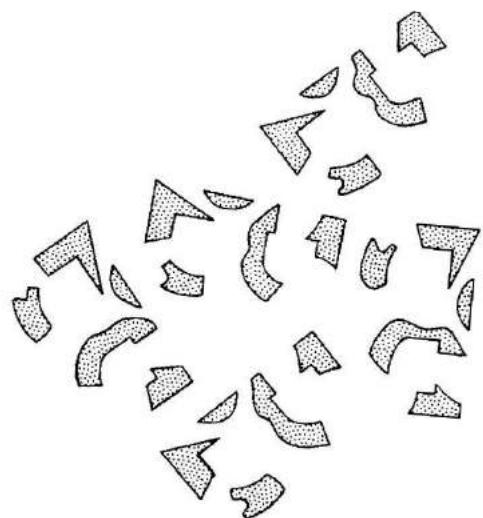
Gestalt Laws: Continuity

The presence of an occluding surface (the elements of which can be grouped together and segmented from the rest of the image) enables us to group the blobs into meaningful shapes.

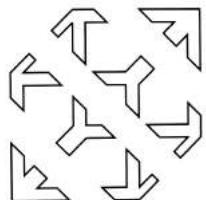


Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Continuity



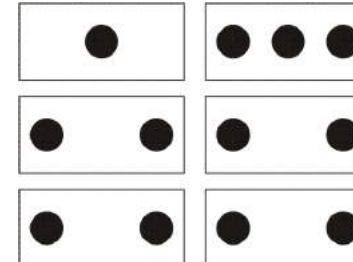
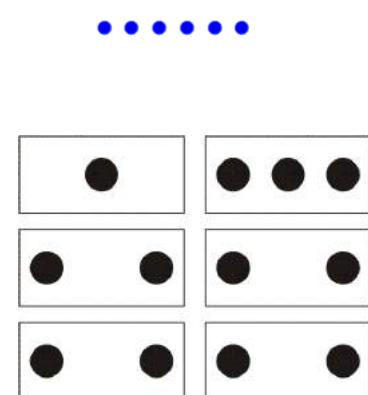
Seen as a collection of unrelated blobs.



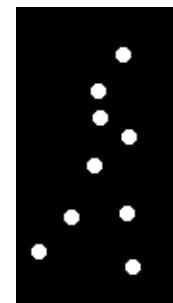
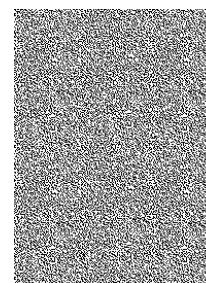
Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Common Fate

Elements that have coherent motion tend to be grouped together.



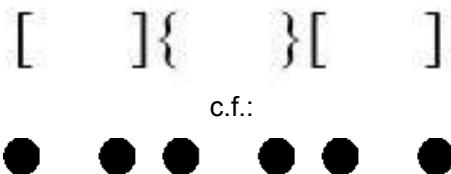
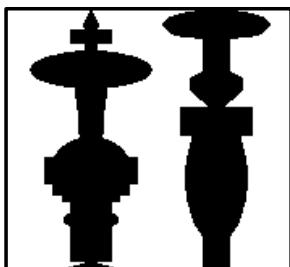
Kolers & Pomerantz, 1971



Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Symmetry

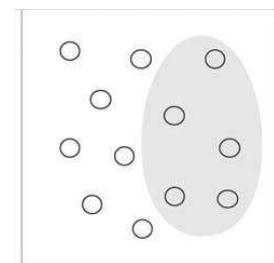
Elements that form symmetric groups tend to be grouped together.



Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Common Region

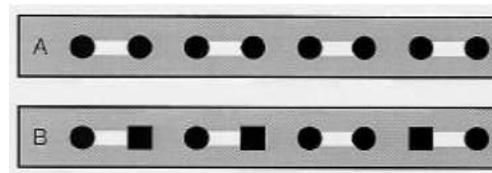
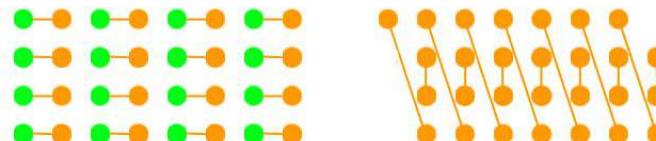
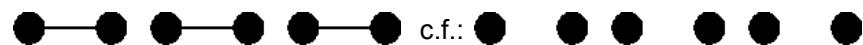
Elements that lie inside the same closed region tend to be grouped together.



Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Connectivity

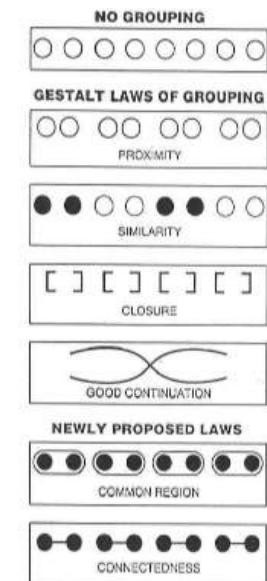
Elements that are connected by another element tend to be grouped together.



Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Summary

- | | | |
|---|---------------------|---------------------------|
| A | • • • • • • • | No Grouping |
| B | • • • • • • | Proximity |
| C | • • • • • • • | Similarity of Color |
| D | • • • • • • • | Similarity of Size |
| E | — — — — | Similarity of Orientation |
| F | • ↓ • ↓ • ↑ • ↓ • ↑ | Common Fate |
| G | | Symmetry |
| H | | Parallelism |
| I | | Continuity |
| J | | Closure |



Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws: Summary

Many other bottom-up cues have been shown to effect segmentation!

Not all are clearly distinct, e.g.:

Law of proximity = Law of Similarity for location

Law of Common Fate = Law of Similarity for motion

No clear rules about what happens when multiple Gestalt principles are in conflict, e.g.:

Law of proximity vs Law of common region (see earlier)

Law of proximity vs Law of connectivity (see earlier)

Law of proximity vs Law of symmetry (see earlier)

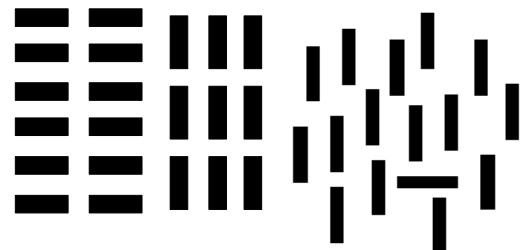
Law of similarity vs Law of continuity:



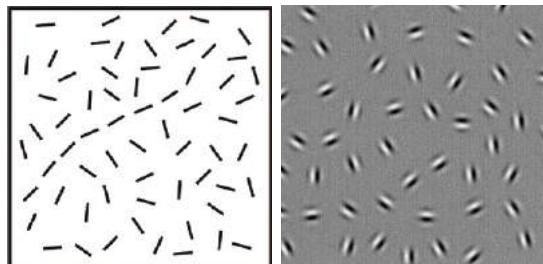
Computer Vision / Low- & Mid-Level Vision / Biological

Gestalt Laws via V1 Lateral Connections

Lateral inhibitory connections, give rise to texture segmentation and pop-out
= Gestalt Law of similarity



Lateral excitatory connections, give rise to contour integration
= Gestalt Law of continuity



Computer Vision / Low- & Mid-Level Vision / Biological

Content

• Border Ownership

- effects on perception
- neural implementation in V2

Computer Vision / Low- & Mid-Level Vision / Biological

Object Segmentation: border ownership

In most tasks we want to group elements into objects.

An object consists of one or more surfaces plus a boundary.

The boundary is “owned” by the object.

The background appears borderless (and hence shapeless).

e.g. the tennis player’s leg has a border, but the grass does not and appears to continue behind the leg:



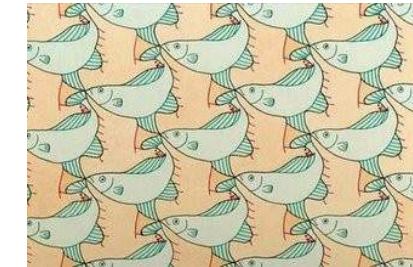
Computer Vision / Low- & Mid-Level Vision / Biological

Object Segmentation: border ownership



Where is the arrow?

Difficult to spot as letters are seen as foreground.

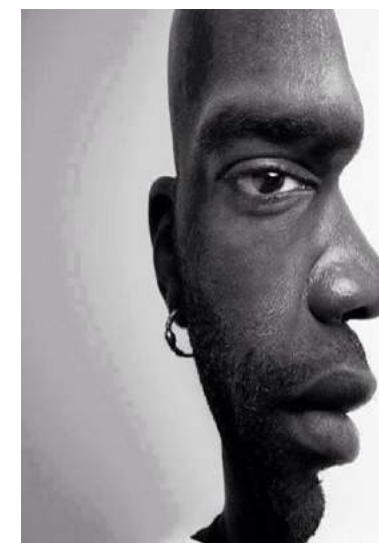


Birds or fish?

Can't see both at the same time, as border can only be owned by one or the other interpretation.

Computer Vision / Low- & Mid-Level Vision / Biological

Object Segmentation: border ownership



You see the face as a profile (looking right) OR a frontal view (looking at you), and you can switch your perception between these different interpretations.

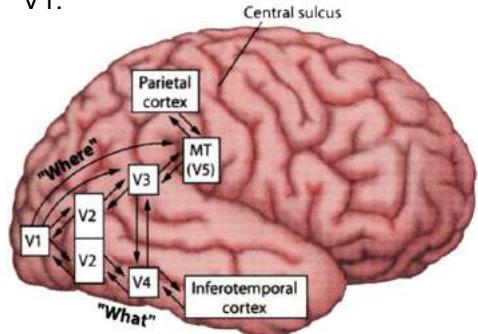
However, you see only one of these two interpretations at any one time.

When you see the profile view the right-hand border is owned by the face, when you see the frontal view the border seems to be owned by a white occluding surface.

Computer Vision / Low- & Mid-Level Vision / Biological

V2 Border-ownership cells

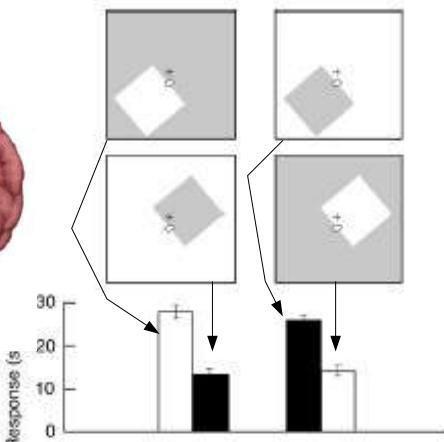
V2 is the cortical area which comes next in the hierarchy to V1.



e.g. This cell is selective for a particular edge orientation, but only responds strongly when the perceived figure is to the lower left of its RF.

Computer Vision / Low- & Mid-Level Vision / Biological

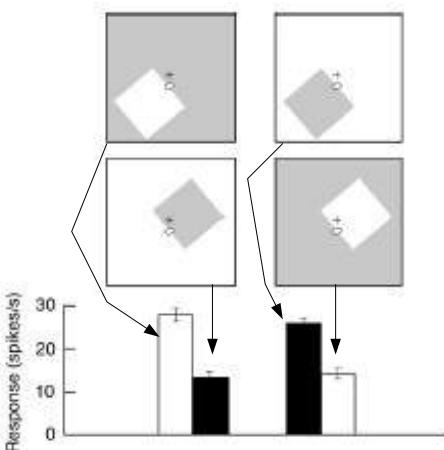
V2 contains cells that encode border-ownership.



Border-ownership via V2 Lateral Connections

The response of V2 border-ownership cells is influenced by information outside the cRF
i.e. border-ownership is determined using the cell's non-classical RFs

V2 contains cells that encode border-ownership.



Computer Vision / Low- & Mid-Level Vision / Biological

Summary

Cortical Visual system is very complex

Consists of hierarchies of processing stages (cortical areas)

1st stage is V1

2nd stage is V2

Computer Vision / Low- & Mid-Level Vision / Biological

Summary

cRFs (pattern of feedforward connections)

In V1:

large variety (colour, orientation, motion, frequency, disparity,...)

mapped across cortical surface

- a patch of V1 represents a particular spatial location
- neighbouring patches represent neighbouring locations

A hypercolumn is a patch which contains all the RFs for the same location

Gabor functions

- provide good model of orientation selectivity (convolution of an image with a Gabor performs edge detection)
- provide efficient codes for natural images

Computer Vision / Low- & Mid-Level Vision / Biological

Summary

ncRFs (pattern of lateral, and feedback, connections)

enables stimuli outside a neuron's cRF to modulate its response

nodes with similar cRFs at neighbouring locations connected via:

- inhibitory connections (generate pop-out, texture segmentation)
- excitatory connections (generate contour enhancement)

increases salience of locations where image changes

Computer Vision / Low- & Mid-Level Vision / Biological

Summary

Mid-Level Vision = grouping image elements together

Which elements are grouped together is influenced by:

- prior experience and object knowledge (top-down influences), and
- Gestalt Laws applied to the image properties (bottom-up influences)

ncRFs in V1 and V2:

- implement some Gestalt principles and produce border-ownership.

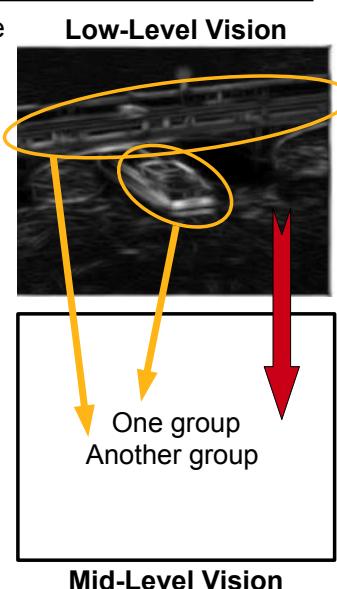
Computer Vision / Low- & Mid-Level Vision / Biological

This Week

group together those elements of an image that “belong together”, and segment these elements from all others.

Methods:

- Thresholding
 - morphological operators
- Region-based
 - region growing
 - region merging
 - split and merge
- Clustering
 - k-means clustering
 - hierarchical clustering
 - graph cutting
- Fitting
 - Hough transform
 - active contours



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Features

Which elements “belong together”? (i.e. lie on the same object)

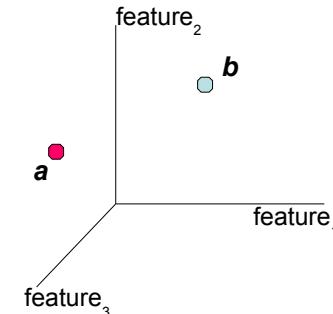
In computer vision many features are used (individually or in combination):

- location (= Gestalt law of proximity)
- colour / intensity (= Gestalt law of similarity)
- texture (= Gestalt law of similarity)
- depth
- motion (= Gestalt law of common fate)
- not separated by contour (= Gestalt law of common region)
- form a known shape when assembled (top-down)
- CNN features

Which feature(s) work best will depend on task

Feature Space

We can think of each image element being a point in feature space. Similarity (and hence grouping) will be determined by the distance between points in this feature space.



a and **b** are vectors of feature values for two elements

$$\mathbf{a} = (a_1, a_2, a_3)$$

$$\mathbf{b} = (b_1, b_2, b_3)$$

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Feature Space: similarity measures

We can think of each image element being a point in feature space. Similarity (and hence grouping) will be determined by the distance between points in this feature space.

Similarity can be measured as:

$$\text{Affinity} = \exp\left(\frac{-\left(\sum_i (a_i - b_i)^2\right)}{2\sigma^2}\right)$$

$$\text{Cross-correlation} = \sum_i a_i b_i$$

$$\text{Normalised Cross-correlation (NCC)} = \frac{\sum_i a_i b_i}{\sqrt{(\sum_i a_i^2)} \sqrt{(\sum_i b_i^2)}}$$

$$\text{Correlation coefficient} = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{(\sum_i (a_i - \bar{a})^2)} \sqrt{(\sum_i (b_i - \bar{b})^2)}}$$

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Feature Space: similarity measures

We can think of each image element being a point in feature space. Similarity (and hence grouping) will be determined by the distance between points in this feature space.

Alternatively, distance can be measured as:

$$\text{Euclidean distance} = \sqrt{\sum_i^n (a_i - b_i)^2}$$

$$\text{Sum of Squared Differences (SSD)} = \sum_i^n (a_i - b_i)^2$$

$$\text{Sum of Absolute Differences (SAD)} = \sum_i^n |a_i - b_i|$$

Other methods of calculating distance and similarity exist, and the choice of function will affect the segmentation that is produced.

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Feature Space: feature scaling

Different features might be weighted differently in the calculation of distance/similarity:

e.g. for Euclidean distance:

$$\sqrt{w_1(a_1 - b_1)^2 + w_2(a_2 - b_2)^2 + w_3(a_3 - b_3)^2 + w_4(a_4 - b_4)^2}$$

where w_1, \dots, w_4 are weights that set relative importance of parameters

determining the weights that yield the best performance is a non-trivial task.

Feature Space: feature scaling

Features of different types may have different scales

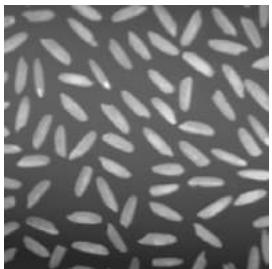
e.g., pixel coordinates on a 2000x2000 pixel image vs. intensity values in the range [0,1].

Problem: Features with smaller scales will appear more similar.

Solution: Scale the features to be in the same range.

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Thresholding



Regions defined by differences in brightness
(could be colour, or output of some filtering process).

The feature space is 1-dimensional.

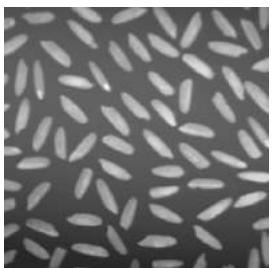
Can segment by thresholding, i.e.

$$l'(x,y) = 1, \text{ if } l(x,y) > \text{thres}$$

$$l'(x,y) = 0, \text{ otherwise}$$

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Thresholding



This results in two groups, foreground and background so is an example of figure-ground segmentation.

How to choose threshold?



150



125



100

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

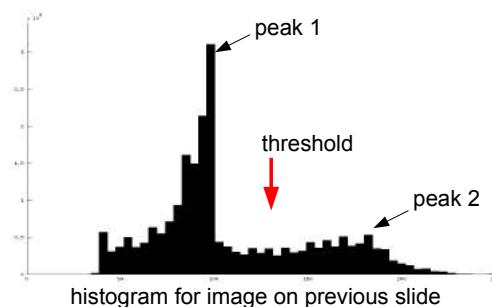
Thresholding: choosing the threshold

1. Use average intensity

2. Plot intensity histogram.

Find peaks

Assuming histogram is bimodal, place threshold in between the peaks



3. Hysteresis thresholding. Define two thresholds (e.g. one each side of valley in histogram)

- Pixels above the high threshold are classified as figure and below the low threshold as background

- Pixels between the low and high thresholds are classified as figure only **if** they are adjacent to other figure pixels

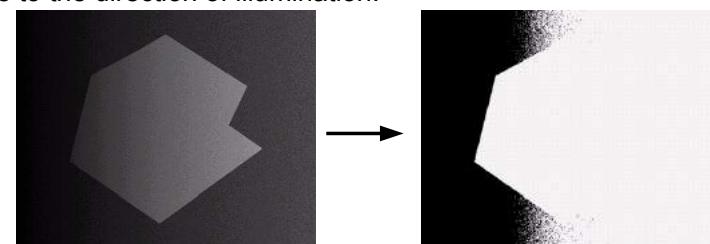
4. Set threshold so that a certain fraction of image is figure (if fraction of image occupied by objects is known)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Thresholding: choosing the threshold

A single threshold is rarely appropriate for a whole image.

Especially, when we have uneven illumination due to shadows or due to the direction of illumination.



It is the local contrast between the object and the background that is important

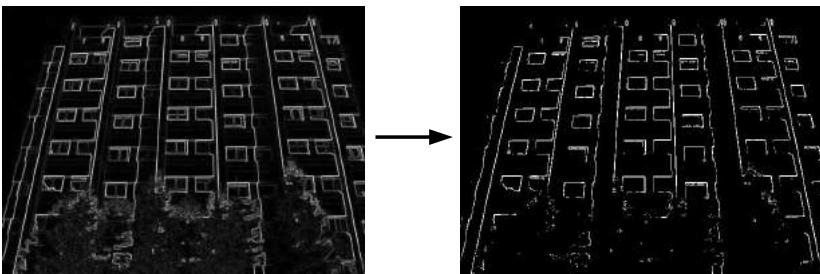
Local thresholding / block thresholding: image split into rectangular blocks and different threshold applied to each block

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Thresholding other features

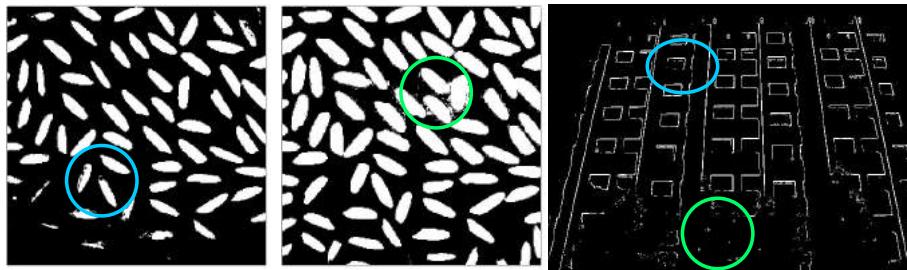
Thresholding might be applied after the image has been processed in some way (preprocessing defines feature space).

e.g. to segment edges from non-edges:



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Thresholding errors



Thresholding often results in

- **missing parts** (e.g. objects/edges with missing pixels)
 - **unwanted parts** (e.g. extra pixels labelled as foreground that are not part of any object/edge)

Morphological Operations

Used to clean up the results of thresholding

Dilation expands the area of foreground pixels (1s) in a binary image (e.g. to fill holes and gaps)

All background pixels that neighbour a foreground pixel are changed from 0 to 1

Neighbourhood defined by a “structuring element”

e.g. a 3x3 pixel square structuring element defines a neighbourhood where each pixel's neighbours are the 8 adjacent pixels horizontally, vertically and diagonally



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Morphological Operations

Used to clean up the results of thresholding

Erosion shrinks the area of foreground pixels (1s) in a binary image (e.g. to remove bridges, branches and small protrusions)

All foreground pixels that **neighbour** a background pixel are changed from 1 to 0.

Neighbourhood defined by a “structuring element”

e.g. a 3x3 pixel square structuring element defines a neighbourhood where each pixel's neighbours are the 8 adjacent pixels horizontally, vertically and diagonally



A 10x10 binary matrix where most elements are 0s. There are several 1s scattered across the grid, forming a sparse pattern. Notable clusters of 1s include a vertical column at x=1, a horizontal row at y=1, a diagonal line from (x=3, y=5) to (x=7, y=9), and a small cluster at (x=8, y=2). A single 0 is located at position (x=5, y=5).

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Morphological Operations



original image dilated and
then eroded ("closed").
Foreground holes are filled



original image eroded and
then dilated ("opened").
Background noise is
removed



Region-based segmentation

A fundamental drawback of thresholding is that it does not take into account spatial information.

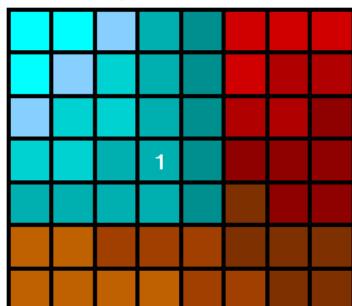
Region based segmentation methods take into account the location of each pixel as well as its intensity, or colour, or texture (or other features or combination of features that are being used to define similarity).

The feature space can be multi-dimensional (following example has 3-d feature space: colour)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region growing

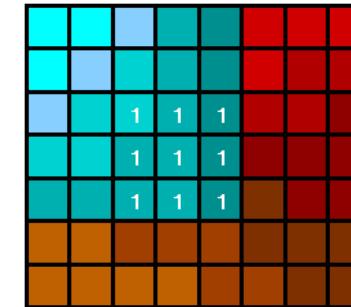
- Start with one “seed” pixel, chosen arbitrarily
 - Give this pixel a label (defining the region it belongs to)
 - Examine all the unlabelled pixels neighbouring labelled pixels
 - If they are within similarity threshold, give them the same region label
- Repeat until region stops growing, then choose another seed pixel which does not yet belong to any region and start again.
- Repeat the whole process until all pixels have been assigned to a region.



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region growing

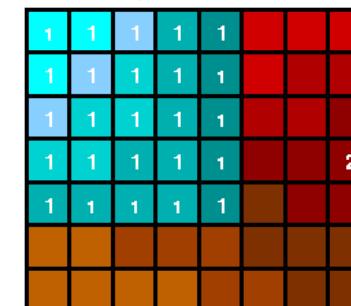
- Start with one “seed” pixel, chosen arbitrarily
 - Give this pixel a label (defining the region it belongs to)
 - Examine all the unlabelled pixels neighbouring labelled pixels
 - If they are within similarity threshold, give them the same region label
- Repeat until region stops growing, then choose another seed pixel which does not yet belong to any region and start again.
- Repeat the whole process until all pixels have been assigned to a region.



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region growing

- Start with one “seed” pixel, chosen arbitrarily
 - Give this pixel a label (defining the region it belongs to)
 - Examine all the unlabelled pixels neighbouring labelled pixels
 - If they are within similarity threshold, give them the same region label
- Repeat until region stops growing, then choose another seed pixel which does not yet belong to any region and start again.
- Repeat the whole process until all pixels have been assigned to a region.



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region growing

- Start with one “seed” pixel, chosen arbitrarily
- Give this pixel a label (defining the region it belongs to)
- Examine all the unlabelled pixels neighbouring labelled pixels
- If they are within similarity threshold, give them the same region label
- Repeat until region stops growing, then choose another seed pixel which does not yet belong to any region and start again.
- Repeat the whole process until all pixels have been assigned to a region.

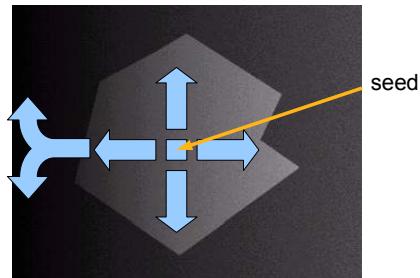
1	1	1	1	1	2	2	2
1	1	1	1	1	2	2	2
1	1	1	1	1	2	2	2
1	1	1	1	1	2	2	2
1	1	1	1	1	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region growing

If similarity is based on intensity, then regions should stop growing at discontinuities in intensity, i.e. edges.

However, region growth may “leak” through a single weak spot in the boundary



Region merging

- Start with each pixel (or small square regions of pixels, e.g. 2x2, or 4x4 pixels) being labelled as separate regions.
- A region's properties are compared with those of an adjacent region
- If they match, they are merged into a larger region and the properties of the new region are computed.
- Continue merging of adjacent regions until a region cannot be merged with any of its neighbours, it is marked ‘final’.
- The whole process repeats until all image regions are marked final.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region merging

- Start with each pixel (or small square regions of pixels, e.g. 2x2, or 4x4 pixels) being labelled as separate regions.
- A region's properties are compared with those of an adjacent region
- If they match, they are merged into a larger region and the properties of the new region are computed.
- Continue merging of adjacent regions until a region cannot be merged with any of its neighbours, it is marked ‘final’.
- The whole process repeats until all image regions are marked final.

1	1	3	4	5	6	7	8
1	1	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region merging

- Start with each pixel (or small square regions of pixels, e.g. 2x2, or 4x4 pixels) being labelled as separate regions.
- A region's properties are compared with those of an adjacent region
- If they match, they are merged into a larger region and the properties of the new region are computed.
- Continue merging of adjacent regions until a region cannot be merged with any of its neighbours, it is marked 'final'.
- The whole process repeats until all image regions are marked final.

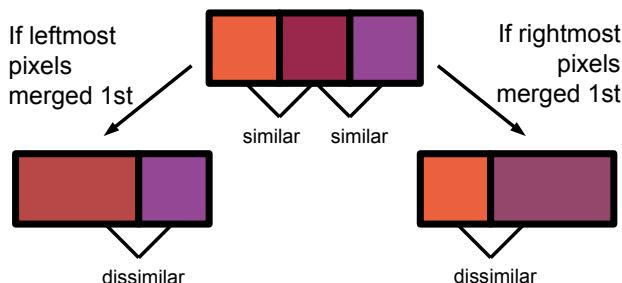
1	1	1	1	1	1	6	7	8
1	1	1	1	1	14	15	16	
1	1	1	1	1	22	23	24	
1	1	1	1	1	30	31	32	
1	1	1	1	1	38	39	40	
41	42	43	44	45	46	47	48	
49	50	51	52	53	54	55	56	

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region merging

The result of region merging usually depends on the order in which regions are merged.

Due to the properties of the merged region being the average of the properties of the constituent parts.



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region splitting and merging

- Start with the whole image labelled as a single region
- For each region:
 - If all pixels are not similar, split the four quadrants into different regions. Continue until each region is homogeneous.
- For each region:
 - Compare to neighbours and merge neighbouring regions which are similar. Continue until no more regions can merge.

1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region splitting and merging

- Start with the whole image labelled as a single region
- For each region:
 - If all pixels are not similar, split the four quadrants into different regions. Continue until each region is homogeneous.
- For each region:
 - Compare to neighbours and merge neighbouring regions which are similar. Continue until no more regions can merge.

1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2
3	3	3	3	4	4	4	4
3	3	3	3	4	4	4	4
3	3	3	3	4	4	4	4
3	3	3	3	4	4	4	4

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region splitting and merging

- Start with the whole image labelled as a single region
- For each region:
 - If all pixels are not similar, split the four quadrants into different regions. Continue until each region is homogeneous.
- For each region:
 - Compare to neighbours and merge neighbouring regions which are similar. Continue until no more regions can merge.

1	1	1	1	1	2	8	5	5
1	1	1	1	1	9	10	5	5
1	1	1	1	1	6	11	7	7
1	1	1	1	1	12	13	7	7
3	3	14	14	4	20	17	17	17
3	3	14	14	21	22	17	17	17
15	15	16	16	18	18	19	19	19
15	15	16	16	18	18	18	19	19

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region splitting and merging

- Start with the whole image labelled as a single region
- For each region:
 - If all pixels are not similar, split the four quadrants into different regions. Continue until each region is homogeneous.
- For each region:
 - Compare to neighbours and merge neighbouring regions which are similar. Continue until no more regions can merge.

1	1	1	1	1	2	8	5	5
1	1	1	1	1	2	8	5	5
1	1	1	1	1	6	11	5	5
1	1	1	1	1	6	11	5	5
3	3	3	3	4	20	17	17	17
3	3	3	3	4	20	17	17	17
15	15	15	15	17	17	17	17	17
15	15	15	15	17	17	17	17	17

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Region-based segmentation

General problems with region-based methods

"Meaningful" regions may not be uniform: e.g. the brightness and colour of a surface may vary due to lighting effects or curvature.

It is very unusual in practice for an image to be composed of uniform regions of similar intensity, or colour, or texture etc.

Example of image segmentation by region growing using colour and texture similarity.



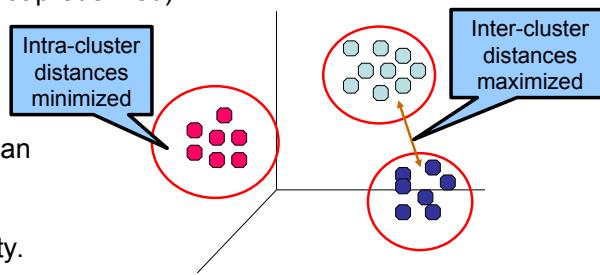
Different colours represent different region labels.

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Clustering

A general class of methods for unsupervised classification of data (i.e. classes are not predefined).

These methods can be used to group image elements based on similarity.



Two main sub-classes of algorithm:

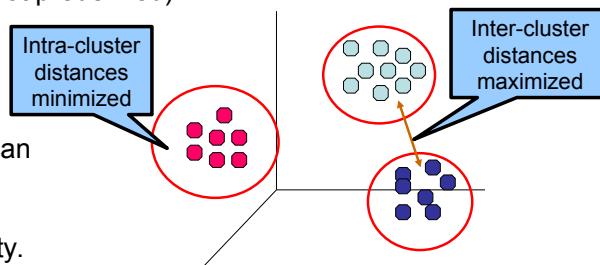
- [Partitional Clustering](#)
 - data divided into non-overlapping subsets (clusters) such that each data element is in exactly one subset
- [Hierarchical clustering](#)
 - A set of nested clusters organized as a hierarchical tree

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Clustering

A general class of methods for unsupervised classification of data (i.e. classes are not predefined).

These methods can be used to group image elements based on similarity.



The feature space is multi-dimensional (following examples have 2-d feature space)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

K-means clustering

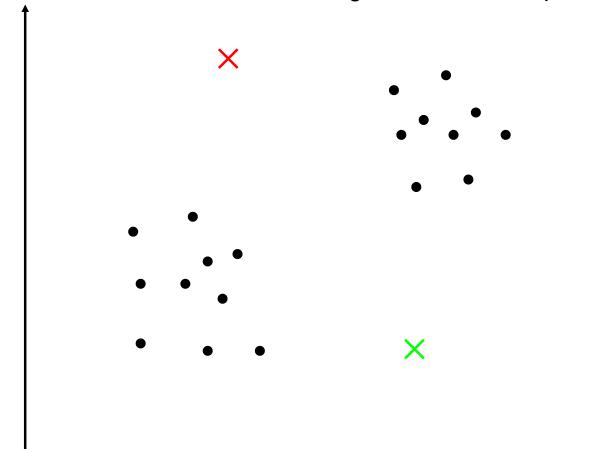
A simple [Partitional Clustering](#) algorithm which assumes that there are k clusters (k is a parameter input to the algorithm)

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

K-means clustering: example

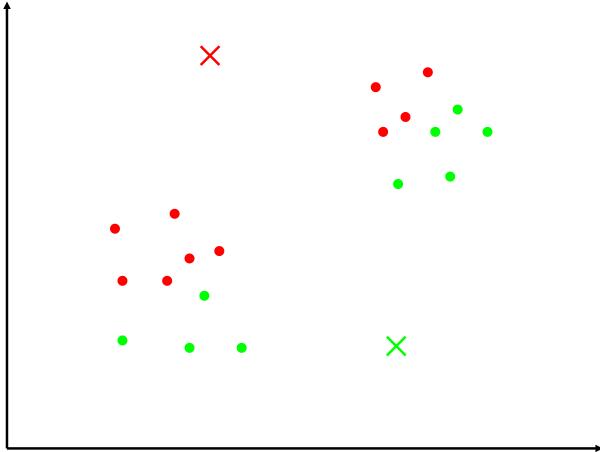
0. **Randomly choose k points to act as cluster centres**
1. Allocate each element to the cluster with the closest centre
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

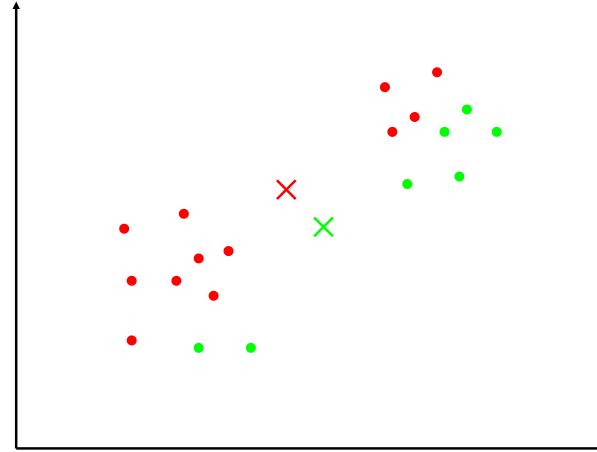
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. **Allocate each element to the cluster with the closest centre**
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1



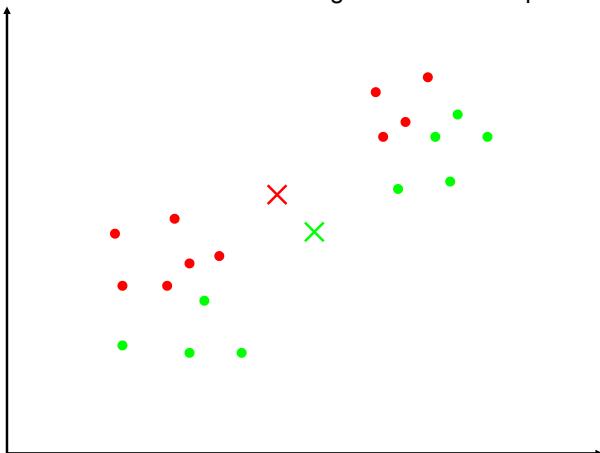
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. **Allocate each element to the cluster with the closest centre**
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1



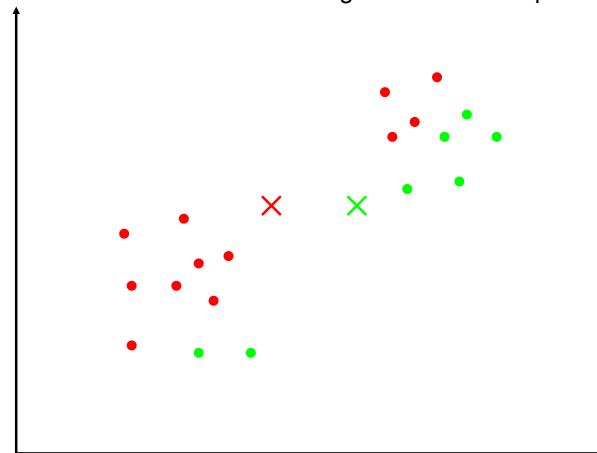
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. **Compute new cluster centres as the mean position of the elements in each cluster**
3. Until the cluster centres are unchanged redo from step 1



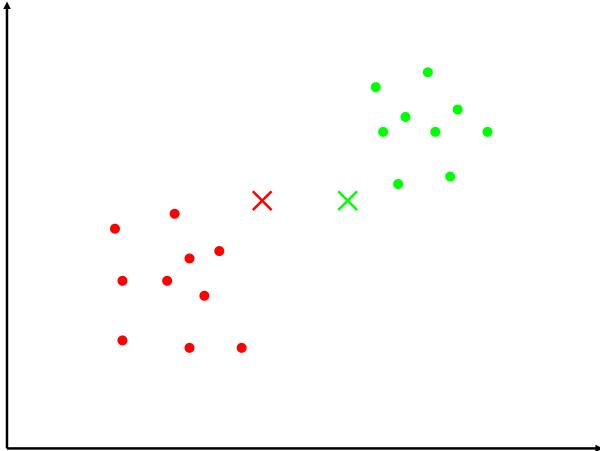
K-means clustering: example

0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. **Compute new cluster centres as the mean position of the elements in each cluster**
3. Until the cluster centres are unchanged redo from step 1



K-means clustering: example

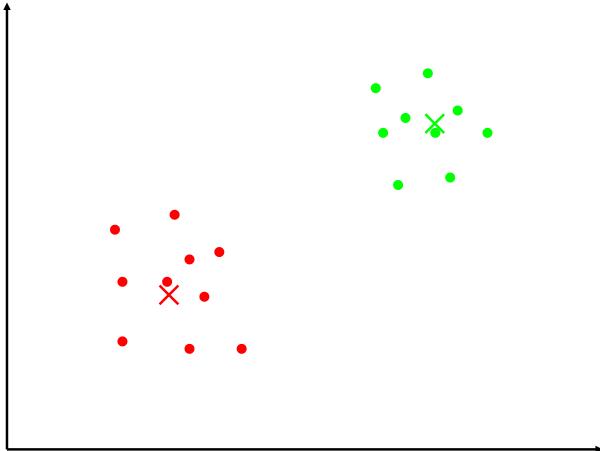
0. Randomly choose k points to act as cluster centres
1. **Allocate each element to the cluster with the closest centre**
2. Compute new cluster centres as the mean position of the elements in each cluster
3. Until the cluster centres are unchanged redo from step 1



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

K-means clustering: example

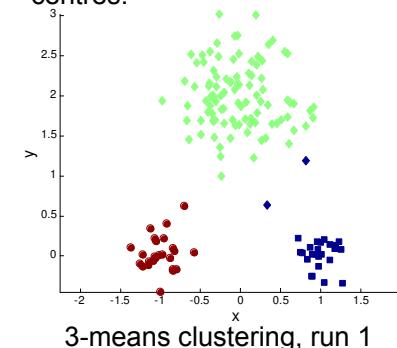
0. Randomly choose k points to act as cluster centres
1. Allocate each element to the cluster with the closest centre
2. **Compute new cluster centres as the mean position of the elements in each cluster**
3. Until the cluster centres are unchanged redo from step 1



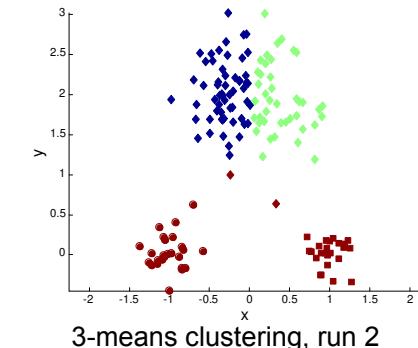
Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

K-means clustering: problems

Results may differ depending on the location of the original cluster centres.



3-means clustering, run 1

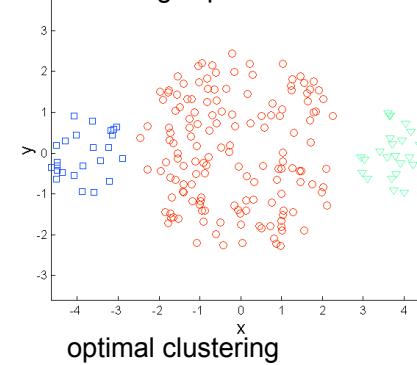


3-means clustering, run 2

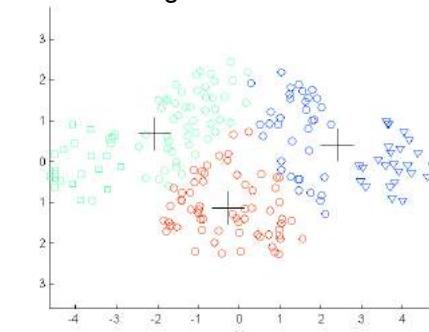
Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

K-means clustering: problems

Clustering is poor if true clusters are of differing sizes.



optimal clustering

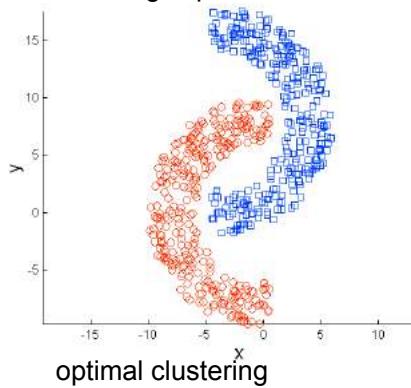


3-means clustering

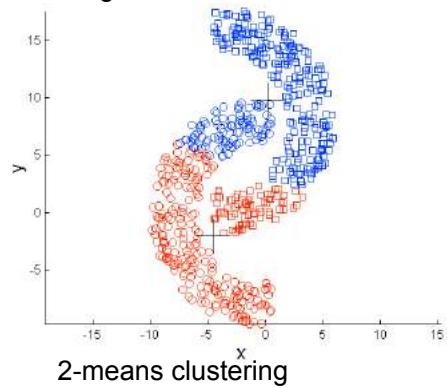
Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

K-means clustering: problems

Clustering is poor if true clusters are not “globular”.



optimal clustering



2-means clustering

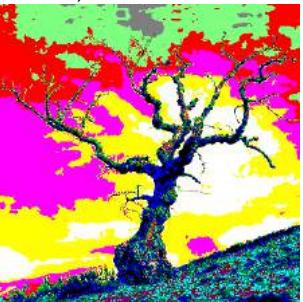
Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

K-means clustering: results

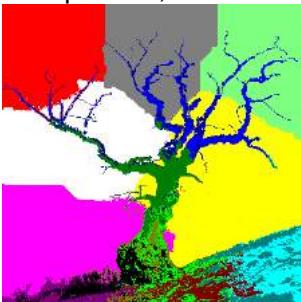
Original image



Clustering using colour alone, k=15



Clustering using colour and position, k=15

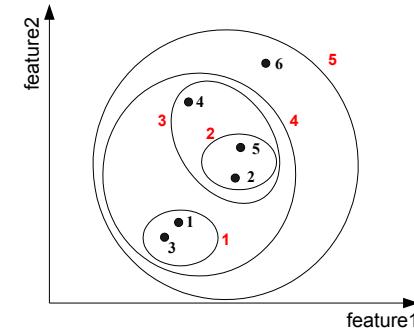


Different colours represent different region labels.

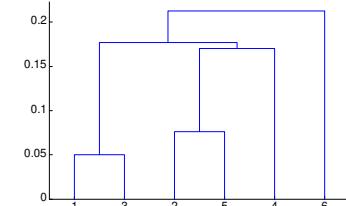
Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Hierarchical Clustering

Produces a set of nested clusters organized as a hierarchical tree



Can be visualized as a dendrogram:



Two sub-classes of methods:

Divisive clustering – the data set is regarded as a single cluster and then clusters are recursively split along best boundary

Agglomerative clustering – each data item is regarded as a cluster, and the clusters are recursively merged by choosing two most similar clusters

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Agglomerative Clustering Algorithm

Basic algorithm:

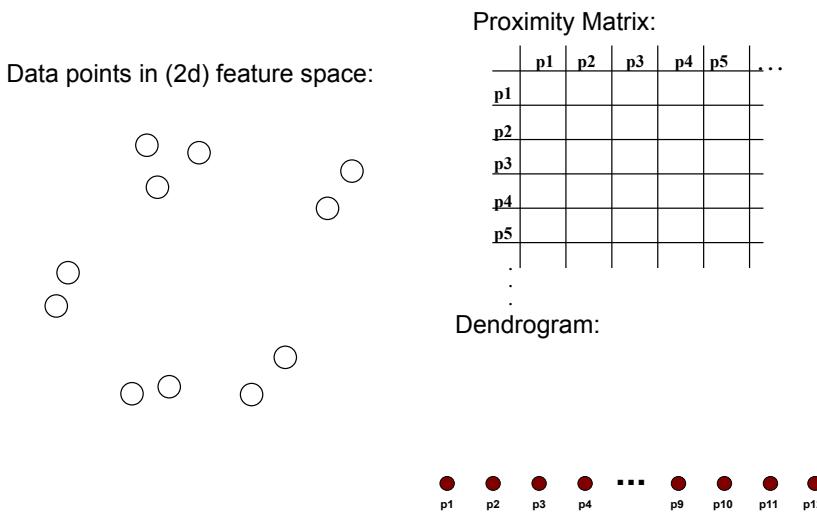
- Let each data point be a cluster
- Compute the proximity matrix
- **Repeat**
 - Merge the two closest clusters
 - Update the proximity matrix
- **Until** only a single cluster remains (or there are a predefined number of clusters, or the two closest clusters are insufficiently similar)

Key operation is the computation of the proximity of two clusters
Different approaches to defining the distance between clusters distinguish the different algorithms

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

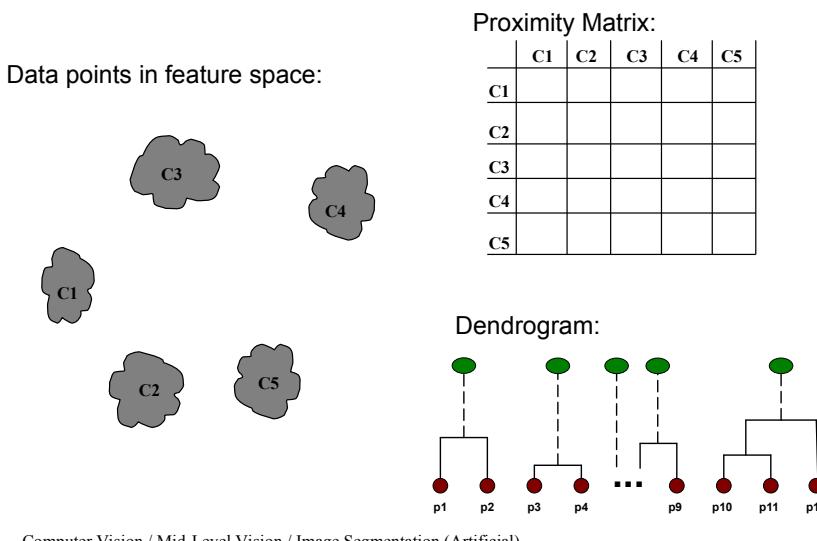
Agglomerative Clustering Algorithm: example

Start with clusters of individual points and calculate proximity matrix



Agglomerative Clustering Algorithm: example

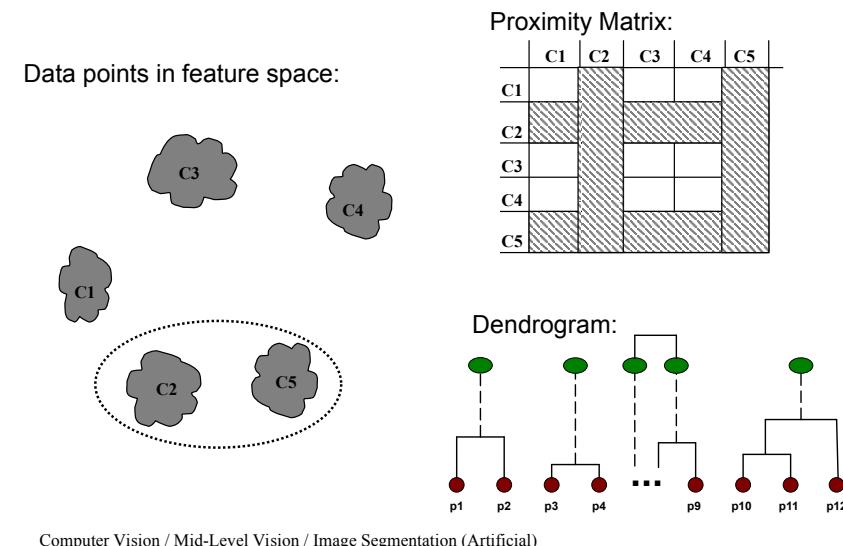
After some merging steps, we have some clusters and a revised proximity matrix



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

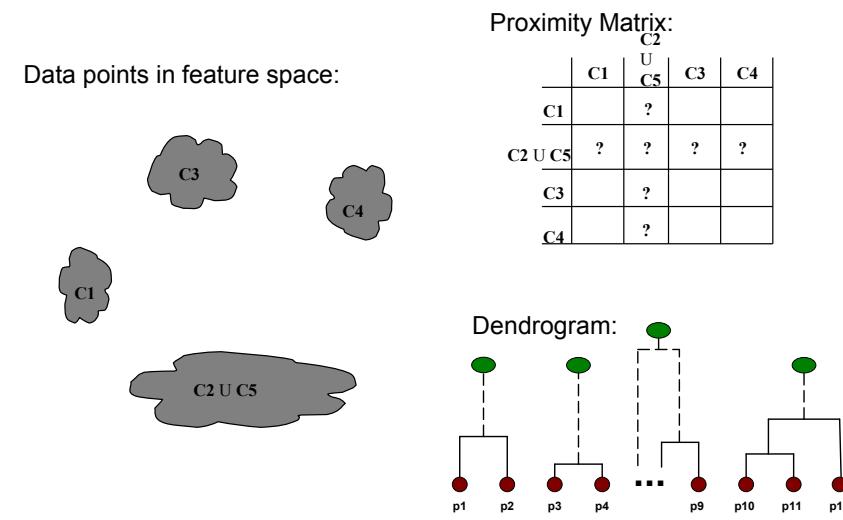
Agglomerative Clustering Algorithm: example

The two closest clusters are now C2 and C5. These need to be merged and the proximity matrix updated



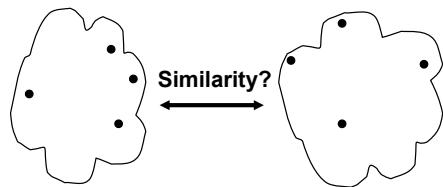
Agglomerative Clustering Algorithm: example

How we update the proximity matrix depends on how we define inter-cluster similarity



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Agglomerative Clustering Algorithm



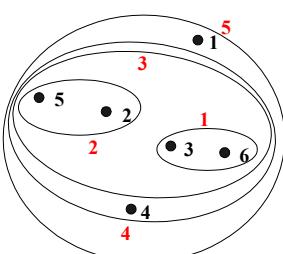
How to Define Inter-Cluster Similarity?

- **single-link** clustering - distance between clusters is shortest distance between elements (MIN distance)
- **complete-link** clustering - distance between clusters is longest distance between elements (MAX distance)
- **group-average** clustering - distance between clusters is average of all distances between elements (AVERAGE distance)
- **centroid** clustering - distance between clusters is the distance between the average of the feature vectors in each cluster

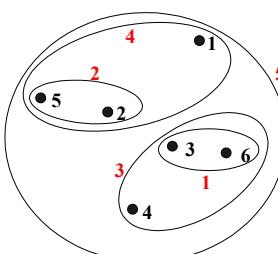
Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Agglomerative Clustering Algorithm

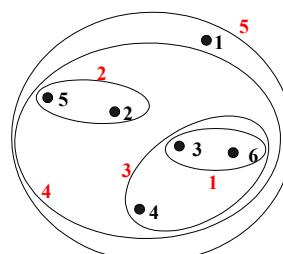
Results will vary depending on the method used to calculate cluster similarity.



MIN distance



MAX distance

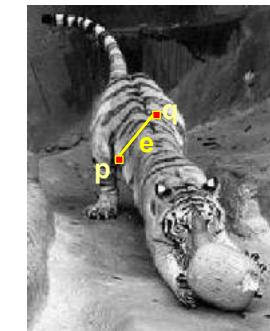
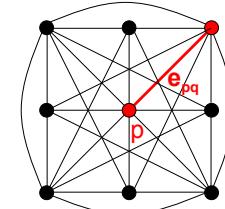


AVERAGE distance

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Segmentation by graph cutting

Feature space can be considered to form a graph $G = (V, E)$



vertices (V) represent image elements (multi-dimensional feature vectors)

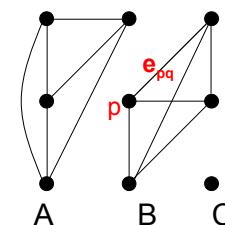
edges (E) connect pair of vertices

each edge encodes the similarity between the two connected elements (e.g. similarity in colour, position, intensity, etc.)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Segmentation by graph cutting

Image segmentation can be viewed as finding an optimal partitioning of the graph (i.e. cutting the graph into disjoint subgraphs).



Graph cutting attempts to produce subgraphs such that the vertices within each subgraph represent elements within the same image region.

Achieved by partitioning the graph so that:

- similarity between subgraphs is minimized (i.e. cut edges are weak)
- similarity within subgraphs is maximized (i.e. subgraphs have strong interior links)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

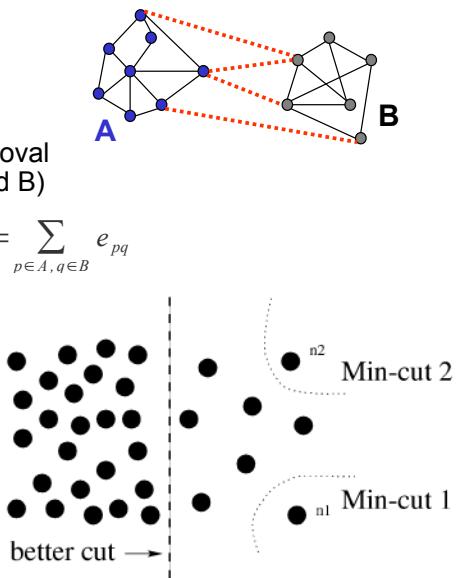
Segmentation by graph cutting

Cutting the graph into disjoint subgraphs will generally require several edges to be cut.

Cost of cutting edges whose removal makes two sets of vertices (A and B) disconnected:

$$cut(A, B) = \sum_{p \in A, q \in B} e_{pq}$$

Finding sets of vertices A and B which give minimum cut cost, will favour cutting small sets of nodes over large sets (cost is proportional to the number of edges in the cut)



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Normalized Cuts (Ncuts)

Bias towards small subgraphs can be overcome by normalising the cut cost by the total strength of all the connections in the two segments.

Normalised cost of cutting edges whose removal makes two sets of vertices (A and B) disconnected:

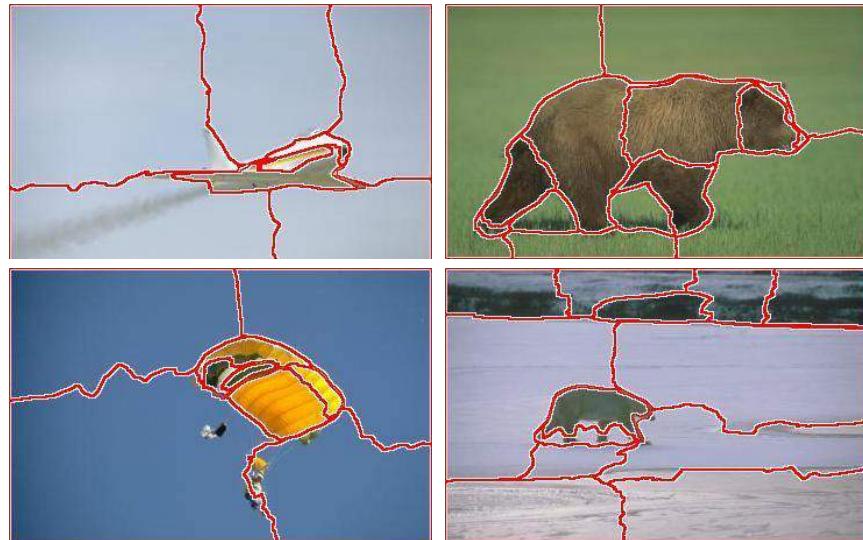
$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$

Where:

$$assoc(A, V) = \sum_{p \in A, q \in V} e_{pq}$$

$assoc(A, V)$ is the total strength of all connections from vertices in set A to all the vertices in the graph

Normalized Cuts (Ncuts): results



Red lines indicate estimated region boundaries.

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Normalized Cuts (Ncuts): problems

- Finding sets of nodes (A and B) which produce the minimum cut is NP-hard:
 - only approximate solutions are possible for real images.
- Bias towards partitioning into equal segments
- Has problems with textured backgrounds (as do most image segmentation algorithms)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Fitting

A class of methods that try to use a mathematical model to represent a set of elements.

Can be used to find the boundaries of objects, and hence, segment the image.

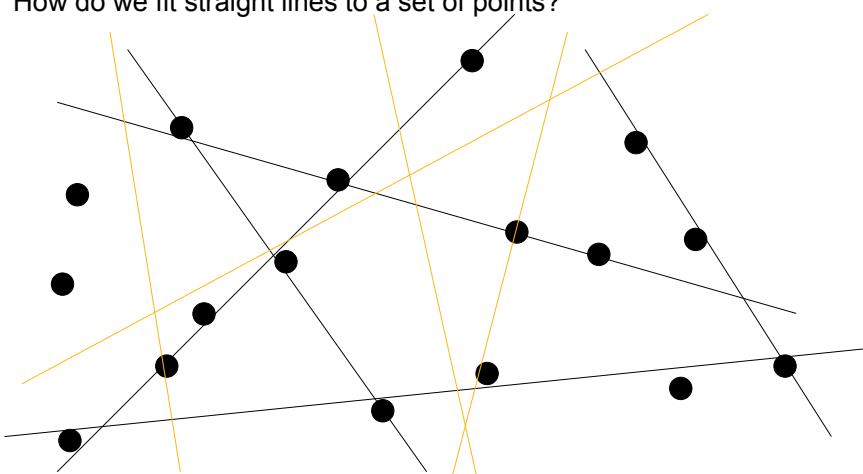
Example:

- A model of the outline of an object that can be rotated and translated to compare it with the image.
- If this model is found to closely fit a set of points or line segments this is unlikely to be due to chance.
- So we represent those elements using the model, i.e. the elements that fit the model are grouped together.

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Fitting: example fitting straight lines

How do we fit straight lines to a set of points?

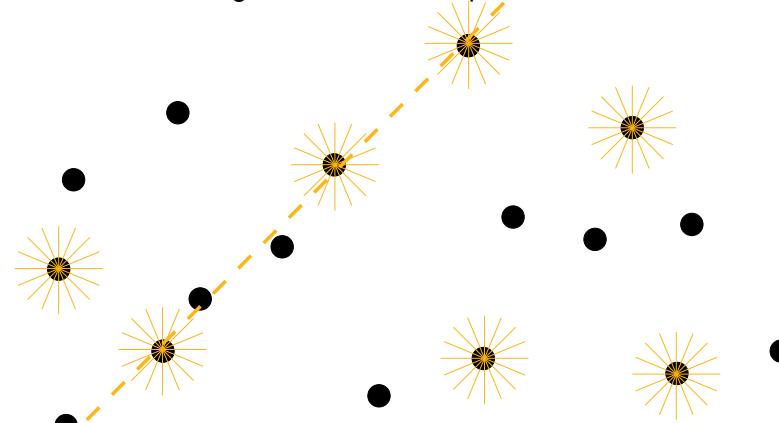


We could superimpose every possible line and see which ones account for the most data.

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Fitting: Hough Transform for fitting lines

How do we fit straight lines to a set of points?



Alternatively, we could see which lines can potentially pass through each point, and count which of these possible lines recur most often.
i.e. each data point votes for all the lines that could pass through it, and lines that have a lot of votes are ones that pass through most points.

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Hough Transform: line parametrisation

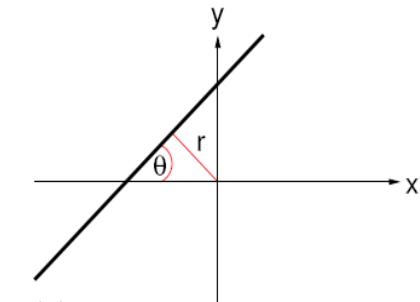
Assume that we want to find elements that form straight lines (i.e. our model is a line)

Representing a line in the usual form, $y = mx + c$, has the problem that m and c go to infinity for vertical lines.

A better choice of parameters for the line is angle, θ , and perpendicular distance from the origin, r .

A line is then the set of points (x, y) such that:

$$y = x \tan(\theta) + \frac{r}{\cos(\theta)} = x \frac{\sin(\theta)}{\cos(\theta)} + \frac{r}{\cos(\theta)}$$
$$y \cos(\theta) - x \sin(\theta) = r$$



We can represent any line by the two parameters θ and r .

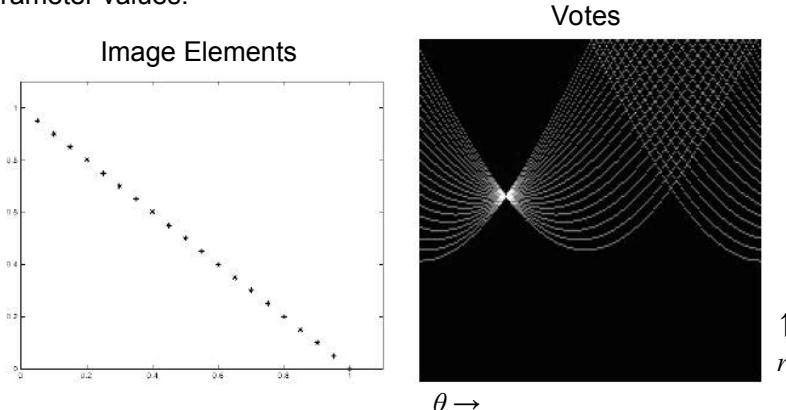
Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Hough Transform: example

Any point (x, y) in an image could potentially form part of a family of lines that pass through this point.

All these lines obey the following relationship: $r = y \cos(\theta) - x \sin(\theta)$

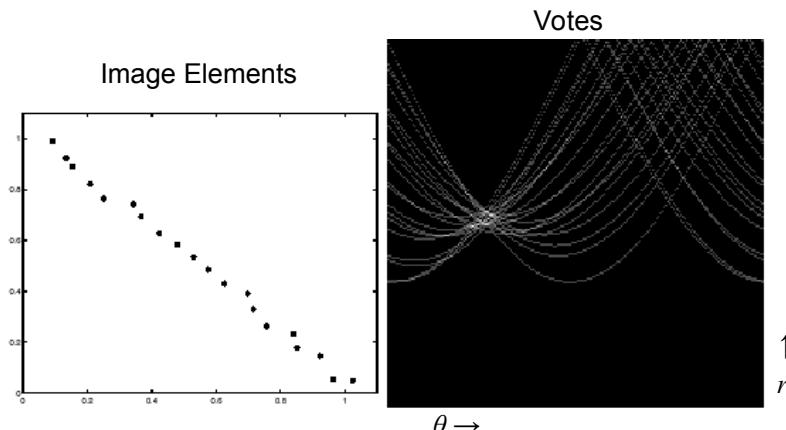
An accumulator array is used to count the votes for each set of parameter values.



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Hough Transform: example

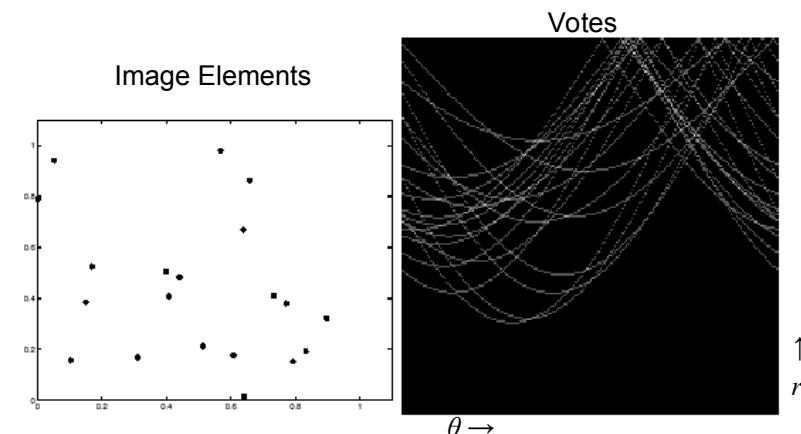
Peak in Hough space becomes blurred as elements are less well aligned



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Hough Transform: example

Multiple, weak, peaks in Hough space result from random, unaligned, points.



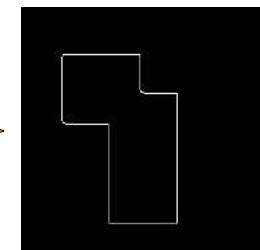
Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Hough Transform: example

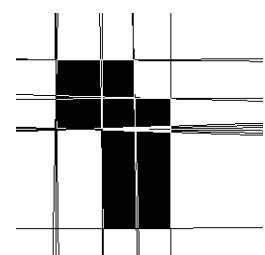
Original Image



Edge detection

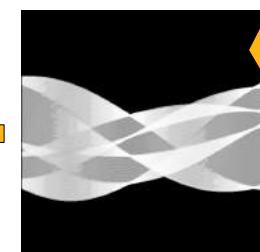


Hough Transform



Find maxima

Plot strongest edges on image



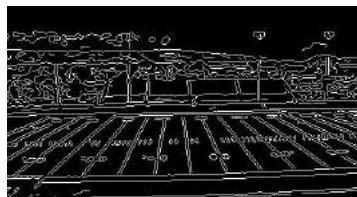
Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Hough Transform: example

Original Image



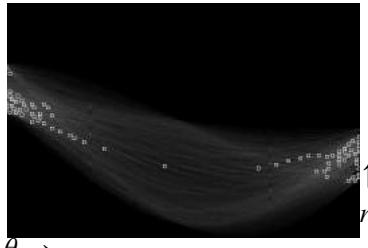
Edge detection



Hough Transform



Find maxima
Plot strongest edges on image



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

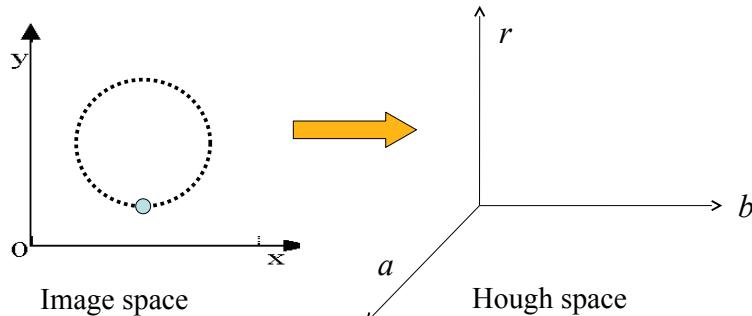
Hough Transform: generalised

The Hough transform can be extended to other shapes that can be expressed parametrically.

e.g. a circle can be described using 3 parameters (a , b , r), that define the centre (a, b) and radius (r) of the circle.

Each point in the image can vote for the (a , b , r) values that encode circles that could potentially pass through that point.

Hence, the accumulator array is 3D in this case.



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

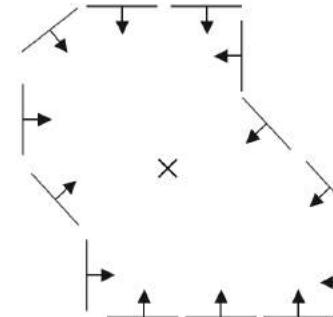
Hough Transform: generalised

The Hough transform can also be extended to arbitrary shapes that can *not* be expressed parametrically.

To do so, make a table that describes the location of each edge pixel in the target shape relative to some reference point.

For each point in the image, assume that it is each edge location in turn, and vote for the location of the reference point.

This is called the Generalised Hough Transform.



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Hough Transform

Advantages

- tolerant of gaps in the edges
- Tolerant to occlusion in the image
- relatively unaffected by noise
- can detect multiple occurrences of a shape in the same pass

Disadvantages

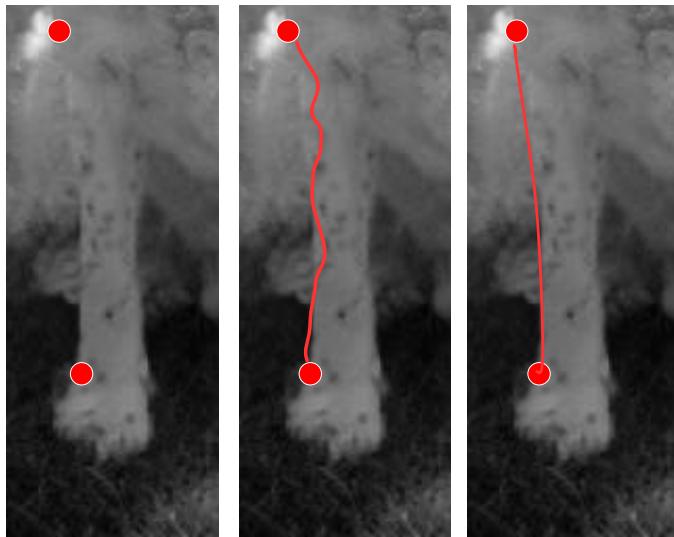
- expensive in terms of memory and computation time
- localisation information can be lost during transformation, e.g. end points of lines are unknown
- peaks in the accumulator can be difficult to locate in presence of noisy or parallel edges
- difficult to determine how coarse or fine the quantization of the accumulator should be: too coarse, and we merge quite different lines; too fine, and noise causes lines to be missed

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Active contours (“snakes”)

Assume we want to find the boundary between these two points.

Which is the best?



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Active contours (“snakes”)

Contour should be:

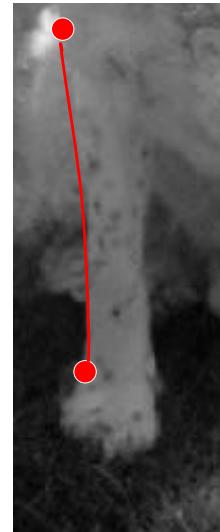
- near the edge
- smooth

A snake is a spline curve that moves so as to minimise “energy”.

Energy is the sum of two terms:

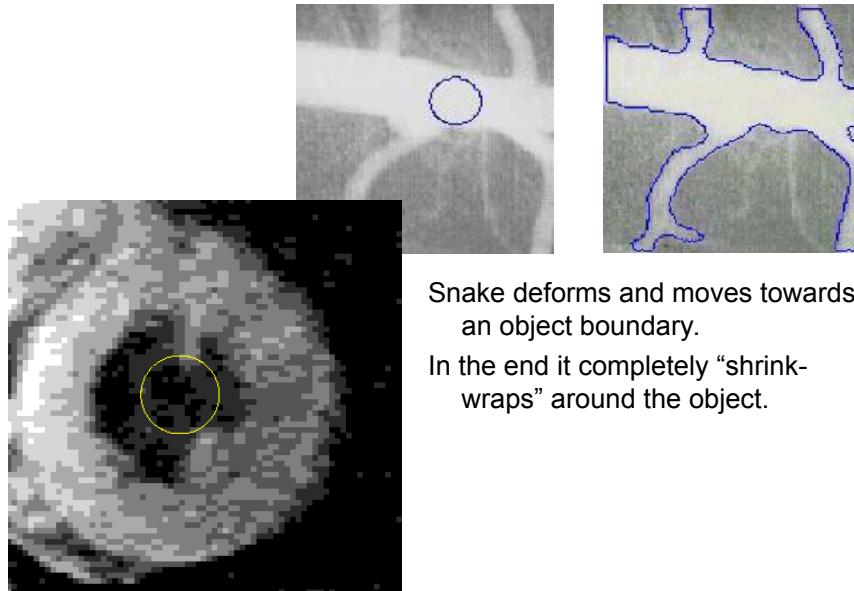
- **internal energy** determined by shape of the snake
 - bending and stretching curve increases energy
- **external energy** determined by proximity of the snake to image features
 - large intensity gradients decrease energy

Minimizing the energy of the snake results in a curve that is short, smooth, and close to intensity discontinuities.



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

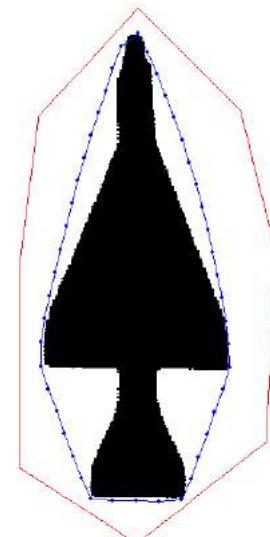
Active contours (“snakes”): examples



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Active contours (“snakes”): problems

- Only works for shapes that are smooth and form a closed contour.
- Extremely sensitive to parameters.
 - parameters control the relative strength of different energy terms (i.e. how big the influence of smoothness, shortness, feature proximity).
- Convergence is dependent on initial position.
 - snake usually placed around object by hand!
- no external force acts on points where there is no intensity gradient, e.g. points which are far away from the boundary.



Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Summary

There are lots of methods of image segmentation.

We can classify segmentation methods in two different ways:

- edge-based vs region-based, or
- model-based vs model-free

Summary

Edge-based methods

attempt to locate the *discontinuities* in image features that define a boundary *surrounding* a region.

Region-based methods

attempt to locate the *similarities* in image features that define the set of pixels *within* a region.

One is the dual of the other.

Segmentations resulting from edge-based methods and region-based methods are not usually exactly the same, and a combination of results may often be a good idea.

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Summary

Edge-based methods

- e.g. Hough transform, active contours, thresholding (applied to the output of an edge detector)
- The approach is to partition an image based on abrupt changes in feature values (i.e. by looking for discontinuities / boundaries)

Region-based methods

- e.g. thresholding (applied to intensity), region growing, region merging, split and merge, k-means clustering, hierarchical clustering, graph cutting
- The approach is to group together image elements that have similar feature vectors (i.e. that look similar)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Summary

Model-based methods

- e.g. Hough transform, active contours
- The approach is to fit the data to a predefined model.

Model-free methods

- e.g. thresholding, region growing, region merging, split and merge, k-means clustering, hierarchical clustering, graph cutting
- The data isn't (explicitly) assumed to fit any particular model, segmentation is based purely on the properties of the image features.

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Computer Vision / Mid-Level Vision / Image Segmentation (Artificial)

Summary

Recall:

Top-down

- elements belong together because they lie on the same object

Bottom-up

- elements belong together because they are similar or locally coherent

Many computer vision systems attempt to perform segmentation before recognition (i.e. they use a bottom-up approach).

However, most of these systems implicitly include some top-down knowledge as the system's designer has made choices about what features to group based on the task they are trying to perform.

Other computer vision systems explicitly use a top-down approach and attempt to fit a model to the data.

Computer Vision (7CCSMCVI / 6CCS3COV)

Recap

- **Image formation**
- **Low-level vision**
- **Mid-level vision**
 - grouping and segmentation of image elements
 - **Biological**
 - bottom-up influences (Gestalt cues)
 - top-down influences (knowledge, expectation, etc.)
 - **Artificial**
 - thresholding, region-based, clustering, fitting
 - **Multi-View Vision**
 - correspondence problem
 - stereo
 - video
- **High-level vision**
 - object recognition

← Today

Multiple Images

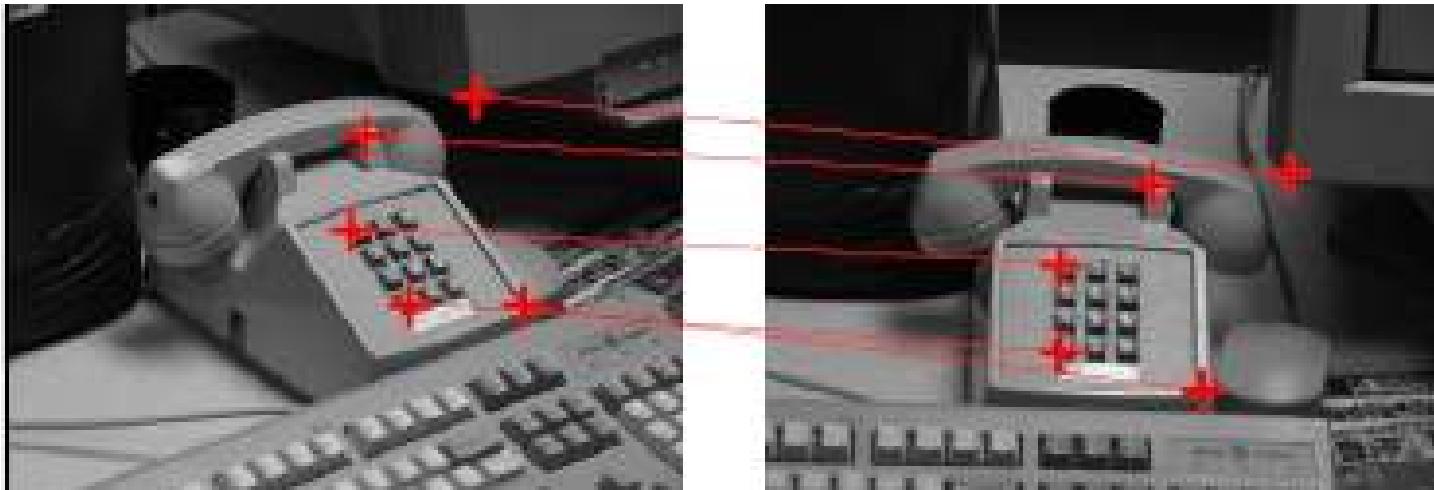
Arise due to:

- multiple cameras (stereo)
 - two, or more, images taken simultaneously by different cameras
- multiple times (video)
 - two, or more, images taken at different times by one camera
- object recognition
 - current image and memorised “training” image(s)

Correspondence Problem

Finding matching image elements across views is fundamental to solving many problems in vision:

- stereo depth recovery
 - finding corresponding points in two images taken by different cameras enables recovery of 3D information
- motion tracking
 - finding corresponding points in two images taken at different times enables estimation of camera and/or object motion
- object recognition
 - finding corresponding points in training and test images enable object recognition



Correspondence Problem

For each selected element in one image, find the corresponding element in the other image.

This is a search problem.

Three main design decisions for correspondence algorithms:

- which elements to match, e.g. intensities, edges, other features.
- how to search for matching elements.
- how to compare elements to confirm/reject match.

Basic requirements to be able to solve the correspondence problem:

1. Most scene points visible in both images
2. Corresponding image regions appear “similar”

Correspondence Problem: problems

Occlusions

some elements may not have a corresponding element due to self-occlusion, occlusion by other object(s), or no longer being “in shot”

False matches

there may be several similar elements, only one of which can be the “true match”

Changing element characteristics

feature values of corresponding elements may differ due to, e.g.:

- change in lighting direction (change in intensity)
- viewpoint differences (change in size and shape)

Large search space

each element in one image has many possible matches in other image.

Methods often employ additional assumptions to constrain search space and resolve ambiguities.

Correspondence Problem: solutions

Algorithms to solve the correspondence problem fall into two classes:

Correlation-based methods

Attempt to establish a correspondence by matching image intensities – usually over a window of pixels in each image

- start from raw image intensity values
- match image windows
- compare them using a similarity measure for intensity values

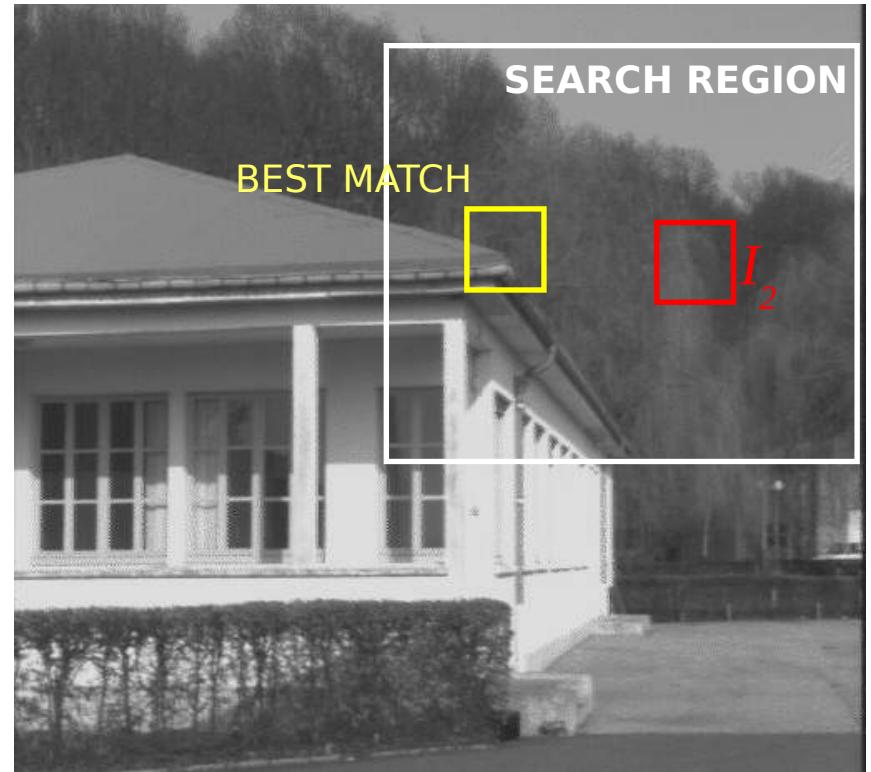
Feature-based methods

Attempt to establish a correspondence by matching sparse sets of image features

- start from image features extracted by preprocessing
- match image features
- compare them using distance between feature descriptors

Correlation-Based Methods

Matching based on correlating pixel values within image regions.



For each region I_1 in image one

For each region I_2 of same size in search area of other image

Compute similarity between I_1 and I_2 .

Repeat for all regions in image two within a search area.

Corresponding point is centre of region giving highest similarity.

Repeat for all regions in image one for which correspondence is required.

Correlation-Based Methods

Need to decide on:

1. Size of correlation window – success depends on window exhibiting a distinctive structure that occurs infrequently in the search region of the other image
 - too **small** a window
 - may not capture enough image structure to be distinctive, and
 - may be too noise sensitive resulting in *many false matches*
 - too **large** a window
 - decreases precision (blurs correspondence map)
 - decreases tolerance to viewpoint
2. Size of search area – full correlation of all pixels is computationally expensive. Therefore usually constrained either:
 - arbitrarily to be around pixel location from which I_1 taken, or by
 - explicit knowledge of task (e.g. using epipolar geometry in stereo correspondence problem [next lecture]).
3. Method used to measure similarity.

Similarity Measures

We can **maximise** the following measures:

Cross-correlation:

$$\sum_{i,j} I_1(i,j) I_2(i,j)$$

If I_1 and I_2 are considered to be vectors in feature space, then the cross-correlation is the dot-product of these vectors

$$I_1 \cdot I_2 = \|I_1\| \|I_2\| \cos \theta$$

Normalised cross-correlation: $\frac{\sum_{i,j} I_1(i,j) I_2(i,j)}{\sqrt{\sum_{i,j} I_1(i,j)^2} \sqrt{\sum_{i,j} I_2(i,j)^2}} = \frac{I_1 \cdot I_2}{\|I_1\| \|I_2\|} = \cos \theta$

Correlation coefficient: $\frac{\sum_{i,j} (I_1(i,j) - \bar{I}_1)(I_2(i,j) - \bar{I}_2)}{\sqrt{\sum_{i,j} (I_1(i,j) - \bar{I}_1)^2} \sqrt{\sum_{i,j} (I_2(i,j) - \bar{I}_2)^2}}$

equals normalised cross-correlation if means are zero

Similarity Measures

We can **minimise** the following measures:

Sum of Squared Differences (SSD): $\sum_{i,j} (I_1(i,j) - I_2(i,j))^2$

Euclidean distance: $\sqrt{SSD} = \sqrt{\sum_{i,j} (I_1(i,j) - I_2(i,j))^2}$

Sum of Absolute Differences (SAD): $\sum_{i,j} |I_1(i,j) - I_2(i,j)|$

In practice, SAD is often used as it is simple and the increased computational costs of using other measures is usually not justified in terms of improved performance.

Correlation-Based Methods: performance

Advantages

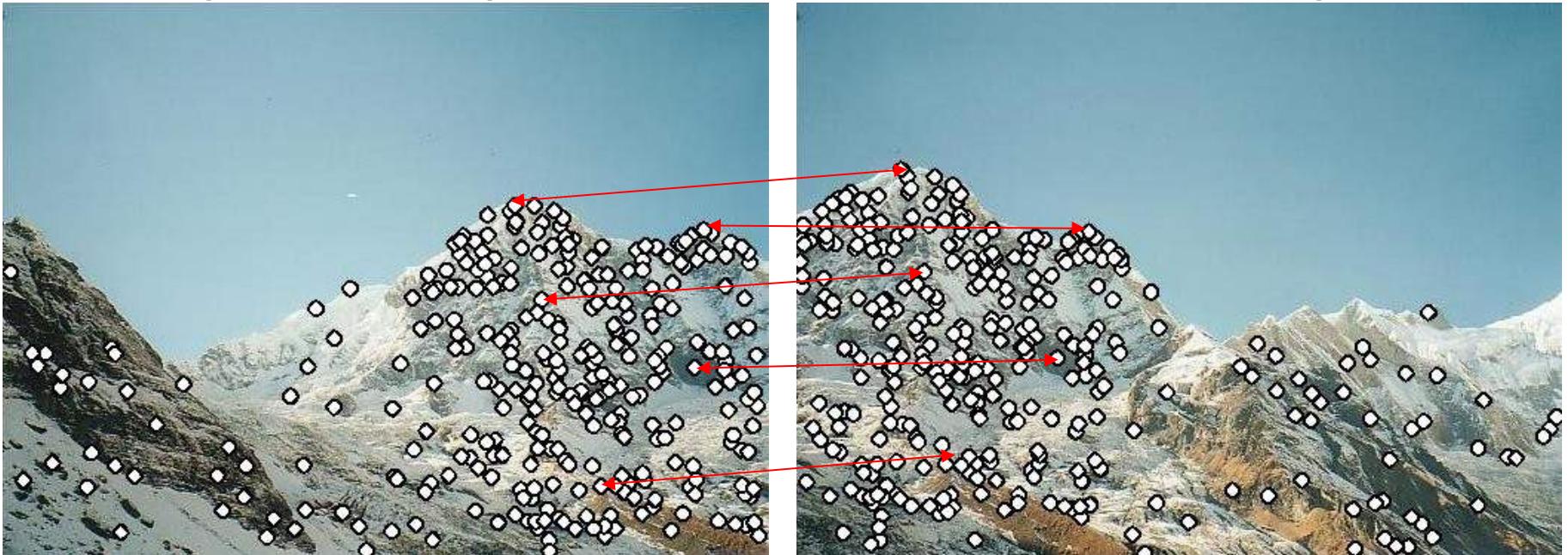
- Easy to implement
- Provides a dense correspondence map (i.e. we calculate correspondence at all points, not just a few)

Disadvantages

- Computationally expensive
- Needs images with lots of distinct intensity patterns to work well
 - regions that are constant or repetitive give many false matches
- Doesn't work well when viewpoints are very different, due to
 - change in illumination
 - changes image intensity values
 - foreshortening
 - changes size/shape/appearance of corresponding regions.

Feature-Based Methods

Matching based on sparse set of features within each image.



Detect interest points in both images

Find corresponding pairs of points:

for each interest point in image one

 for each interest point in search area of other image

 compute similarity between features

 repeat for all interest points in image two within a search area.

 select the interest point in image two that maximizes the similarity measure

 repeat for each interest point in image one

Feature-Based Methods: performance

Advantages

- Relatively insensitive to illumination and size changes
- Less computationally expensive than correlation-based methods
(only need to match selected locations rather than every pixel)

Disadvantages

- Provides sparse correspondence map
 - if correspondence required at intermediate locations this needs to be interpolated
 - sufficient for many applications
- Only suitable when “good” interest points can be extracted from the scene
 - doesn’t perform well with textured/random regions
 - choice of interest points is important

Feature-Based Methods: interest points

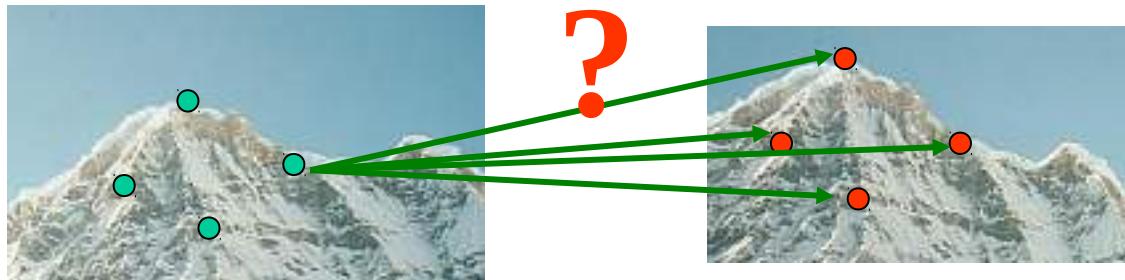
1. Need to detect the *same* point *independently* in both images



We need a repeatable detector

Interest points should be invariant to image scaling, rotation, translation (caused by changing 3D camera viewpoint) and to change in illumination

2. Need to correctly recognize corresponding points in both images



We need a distinctive descriptor

Feature descriptors should be sufficiently complex so that corresponding points can be correctly matched with high probability.

Feature-Based Methods: interest points



Hard to localize
Low information content
BAD interest point

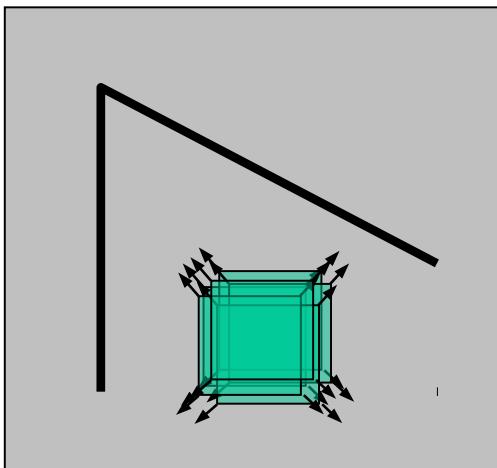
Easy to localize
High information content
GOOD interest point

Interest points: corners

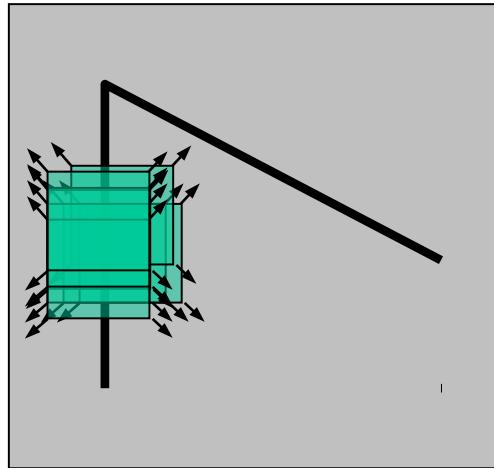
Corners provide repeatable points that are well suited for matching.

A corner = a point where two edges meet

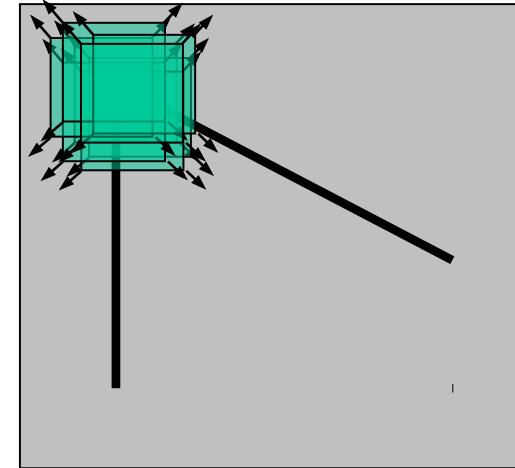
Hence, in the region around a corner, the intensity gradient is high in two directions



“flat”:
no change in all
directions



“edge”:
no change along
the edge direction



“corner”:
significant change
in all directions

Corner Detection

Compute intensity gradients (I_x and I_y) in x and y directions by convolving image with derivative of Gaussian masks

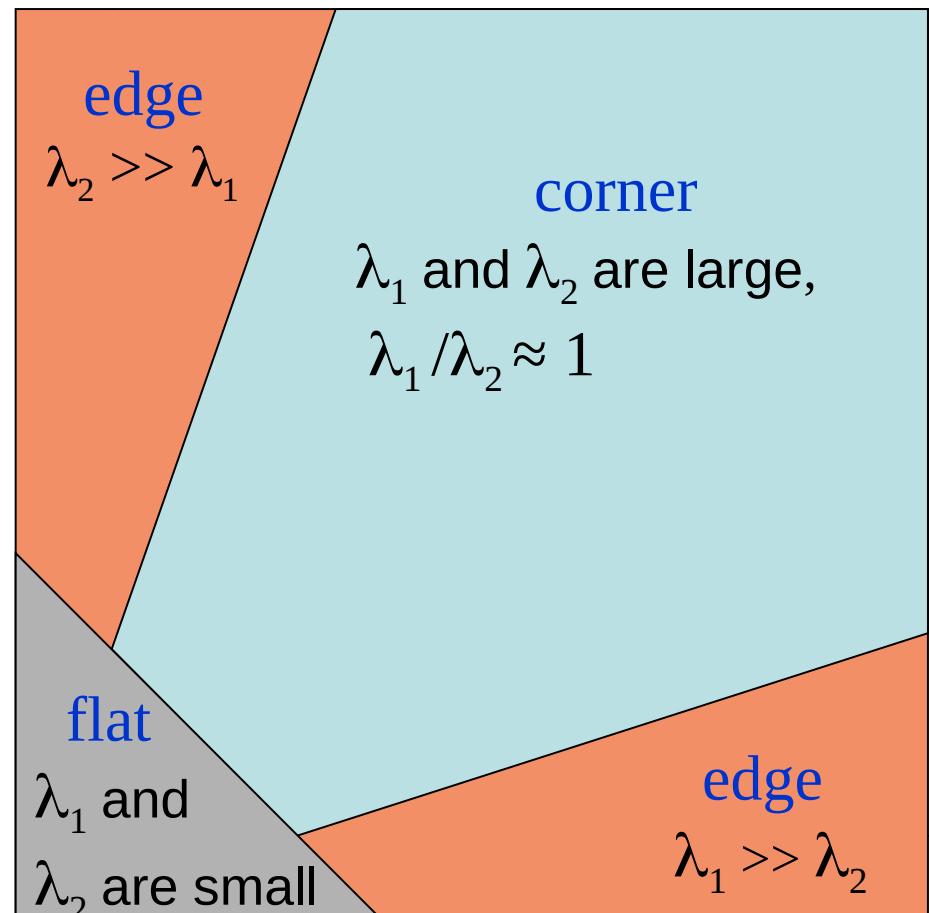
Sum gradients over a small neighbourhood (Gaussian window) around each pixel to generate Hessian matrix H

☞
$$H = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \quad \lambda_2$$

Find eigenvalues, λ_1 and λ_2 , of H .

Eigenvalues correspond to maximum slope of intensity gradient at two orthogonal directions.

Corner is where $\min(\lambda_1, \lambda_2) >$ threshold



Harris corner detector

Avoids calculating eigenvalues. Defines a measure, R, as:

$$R = \det(H) - k (\text{trace}(H))^2$$



$$R = \left[\sum I_x^2 \sum I_y^2 - (\sum I_x I_y)^2 \right] - k \left[\sum I_x^2 + \sum I_y^2 \right]^2$$

($k = 0.04-0.06$ found to work empirically)

R is large for a **corner**

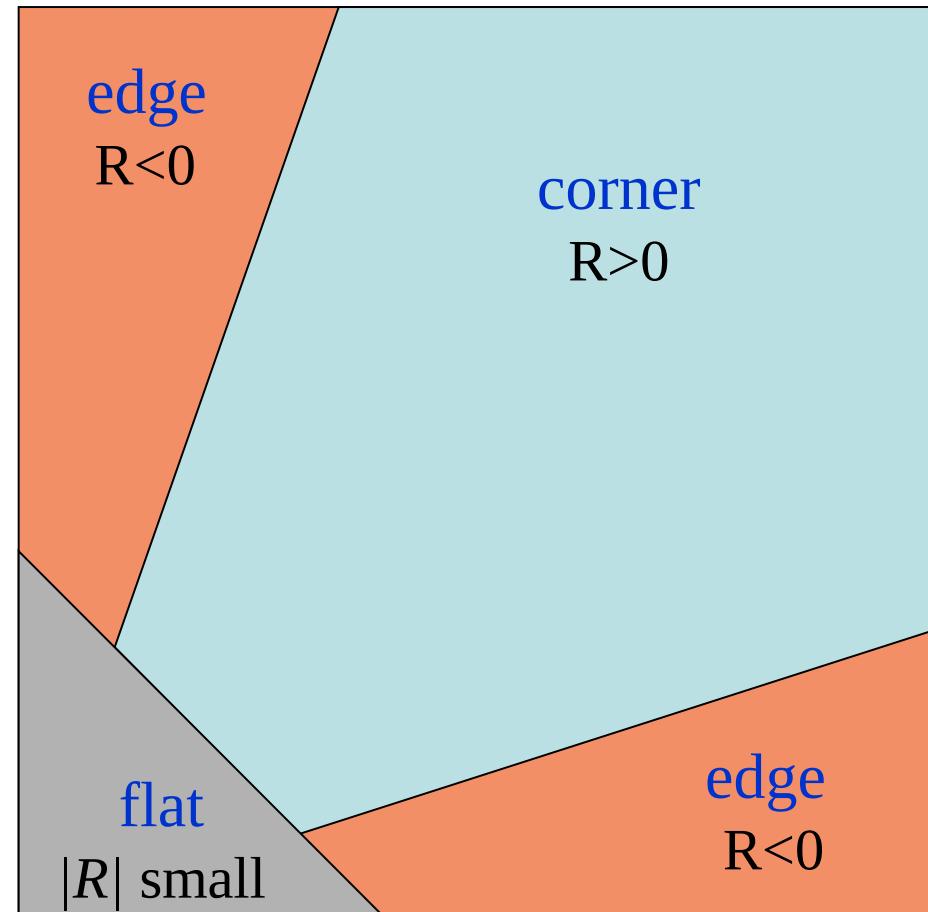
R is negative with large magnitude for an **edge**

$|R|$ is small for a **flat** region

Hence,

Corner is where $R >$ threshold

Take the points of **local maxima** of R as interest points

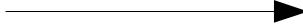


Harris corner detector

Take the points of local maxima of R as interest points.

Use non-maximum suppression:

For each pixel, if it has a neighbour with a larger R value, set its value to 0.



1	2	2	2
0	1	4	3
0	2	2	2
0	1	1	0

0	0	0	0
0	0	4	0
0	0	0	0
0	0	0	0

Non-maximum suppression is commonly used in computer vision, not just for interest point detection

Harris corner detector: algorithm

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x2} = I_x \cdot I_x \quad I_{y2} = I_y \cdot I_y \quad I_{xy} = I_x \cdot I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x2} = G_{\sigma'} * I_{x2} \quad S_{y2} = G_{\sigma'} * I_{y2} \quad S_{xy} = G_{\sigma'} * I_{xy}$$

4. Define at each pixel (x, y) the matrix

$$H(x, y) = \begin{bmatrix} S_{x2}(x, y) & S_{xy}(x, y) \\ S_{xy}(x, y) & S_{y2}(x, y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

$$R = \text{Det}(H) - k(\text{Trace}(H))^2$$

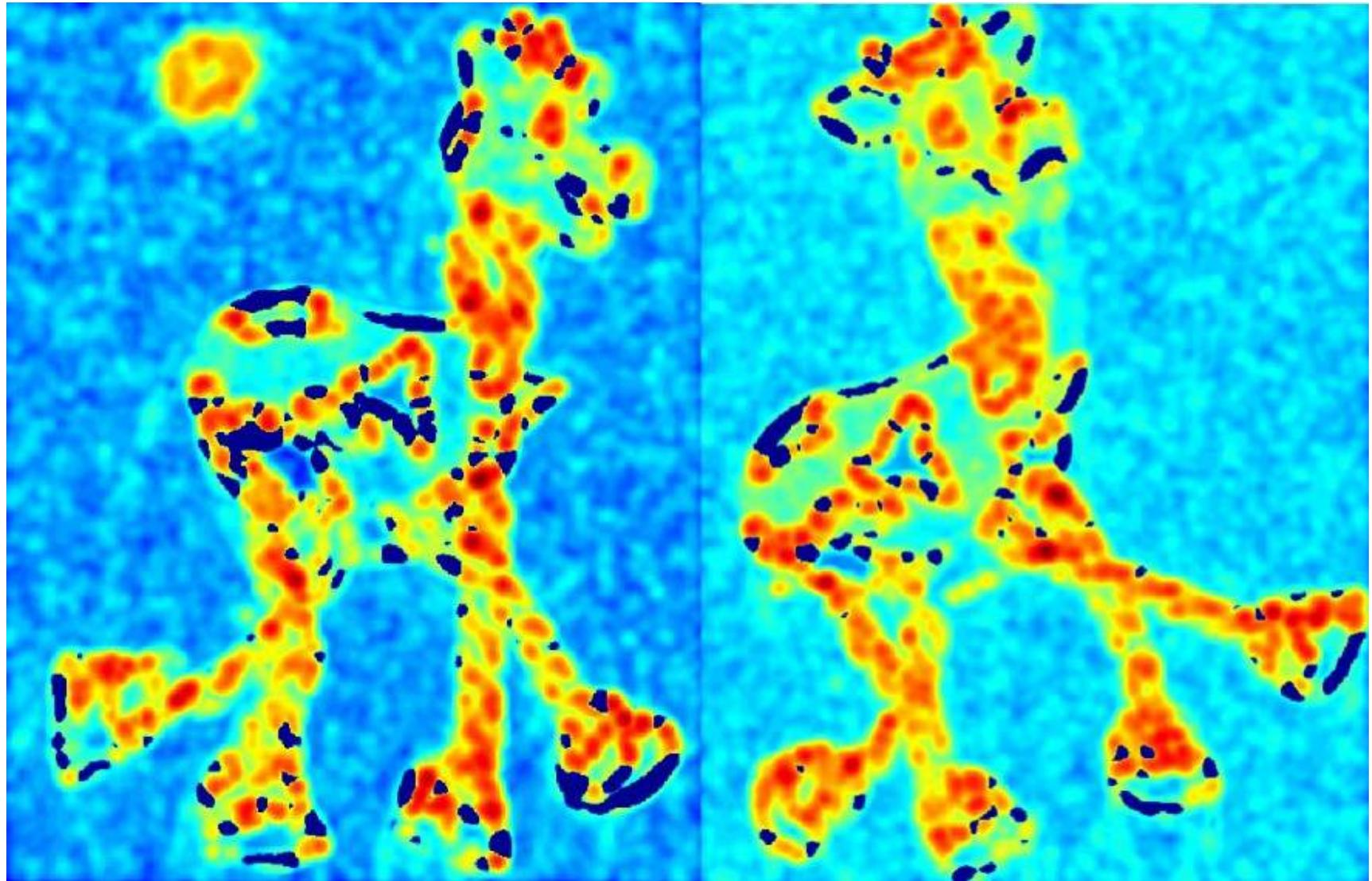
6. Threshold on value of R. Compute nonmax suppression.

Harris corner detector: example



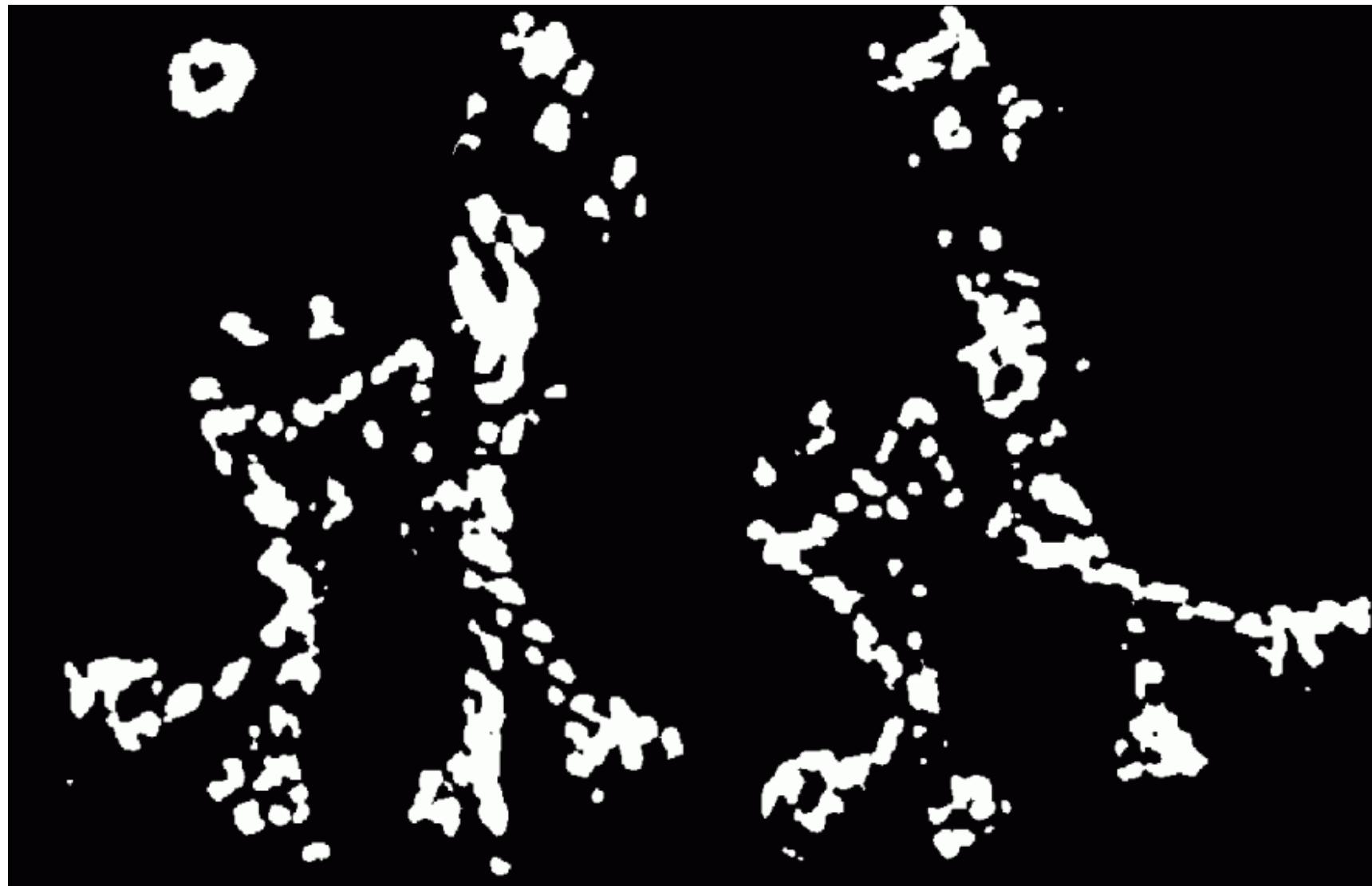
Two images

Harris corner detector: example



Corner response R
red=+ve, black=-ve

Harris corner detector: example



Thresholded R

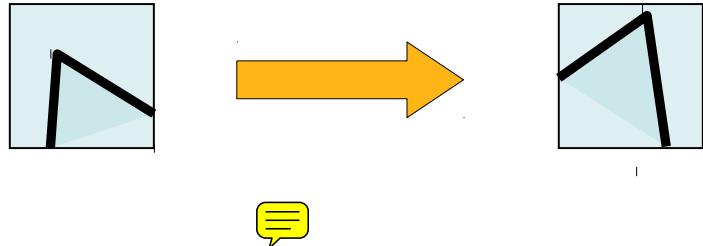
Harris corner detector: example



Local maxima of R

Interest points: corners

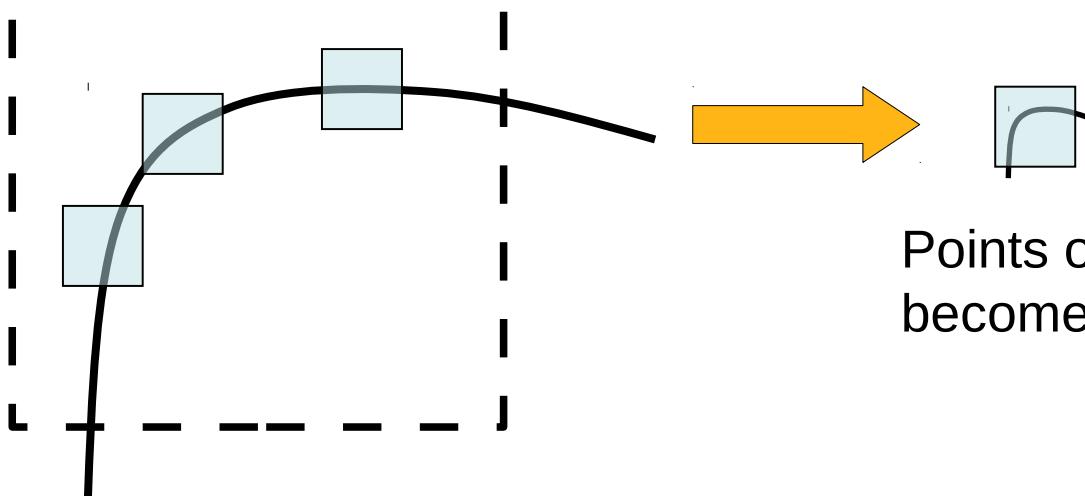
Harris corner detector is translation and rotation invariant



Eigenvectors rotate, but eigenvalues (and R values) remain the same.

Harris corner detector is partly invariant to changes in illumination and to changes in viewpoint.

Harris corner detector is not scale invariant



Points originally classified as edges become corners at coarser scale.

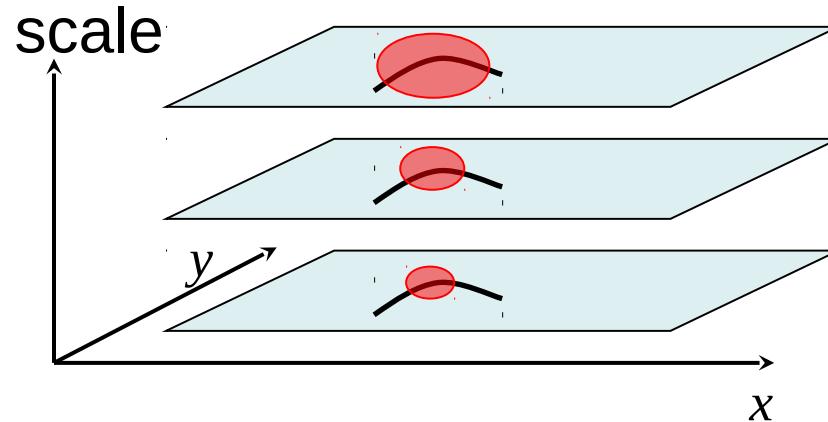
Interest points: scale invariant

To overcome sensitivity to scale, we can perform corner detection across a range of scales using an image pyramid.

Harris-Laplacian

Find local maximum of:

- Harris corner detector in space and scale

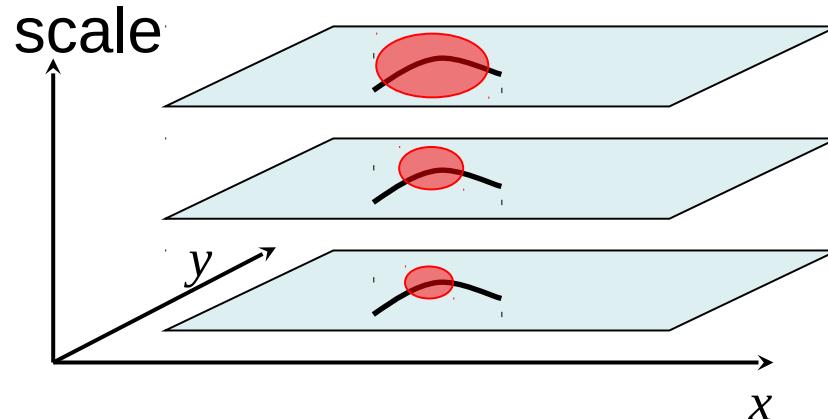


An alternative algorithm for scale invariant interest point detection is the Scale Invariant Feature Transform (SIFT).

SIFT

Find local maximum of:

- Difference of Gaussians in space and scale



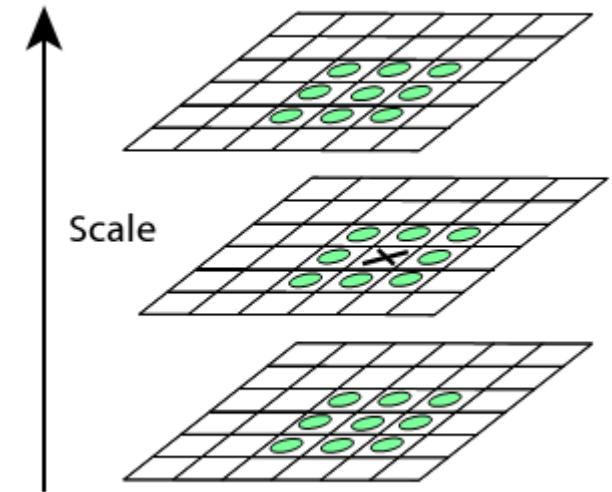
Interest points: scale invariant

Both the Harris-Laplacian and SIFT algorithms search for the maxima of suitable functions (interest point detectors) in scale and in space

This will enable us to find the same interest points independently in two images which differ in scale

SIFT: interest point detection

- Convolve image with Difference of Gaussians (DoG) mask
- Repeat for different image resolutions (i.e. create a Laplacian image pyramid)
- Detect maxima and minima of difference-of-Gaussian across scale space (x selected if larger or smaller than all 26 neighbours in $3 \times 3 \times 3$ neighbourhood)



SIFT: interest point detection

- Keep points with high contrast.
- Keep points with sufficient structure: approach similar to Harris corner detector but using ratio of Trace and Determinant of Hessian matrix.

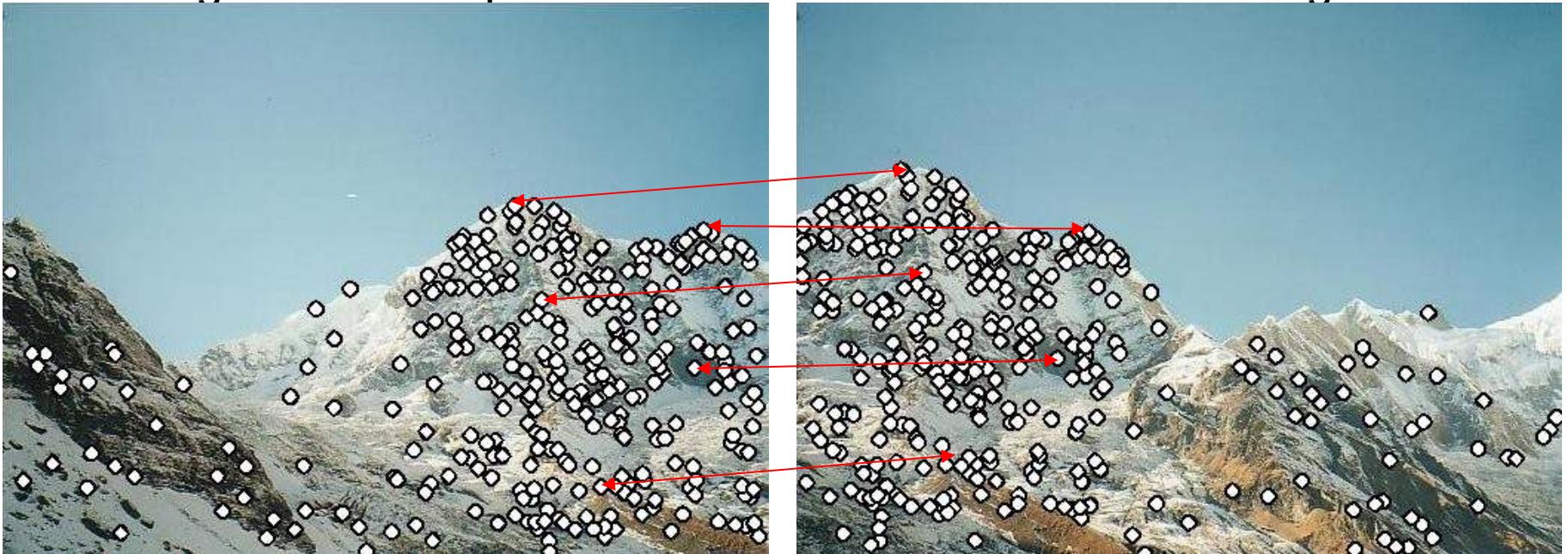
$$\frac{(trace(H))^2}{det(H)} < \frac{(r+1)^2}{r}$$

($r = 10$ found to work empirically)



Feature-Based Methods

Matching based on sparse set of features within each image.



Detect interest points in both images

Harris corner detector, or

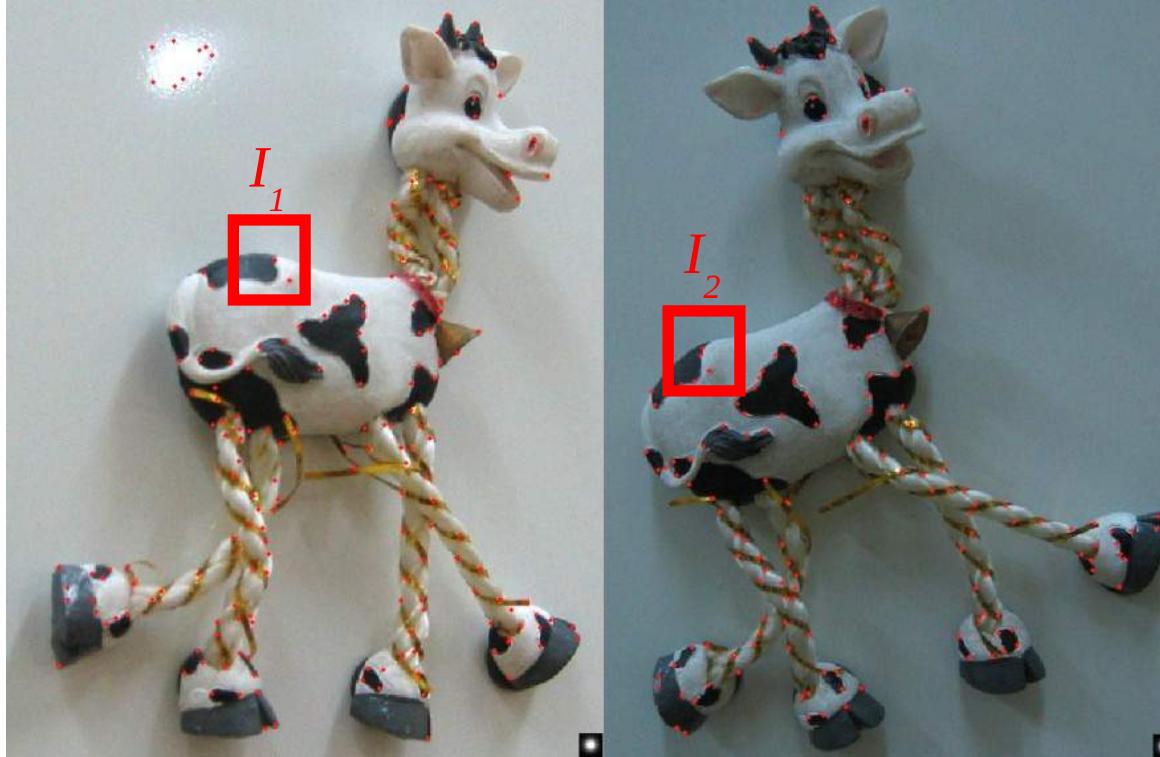
SIFT detector

Find corresponding pairs of points

Requires a measure of similarity between points.

Need a “descriptor” (a list of features) for each interest point.

Harris: feature descriptor and matching



Descriptor: a small window around the interest point (i.e. a set of pixel intensity values).

Similarity measure: Euclidean distance, SSD, SAD, etc.

Robust to translation, but not to rotation, scale, changes in viewpoint or illumination.

SIFT: feature descriptor

Descriptor: the SIFT algorithm specifies a method for deriving a set of features for each interest point.

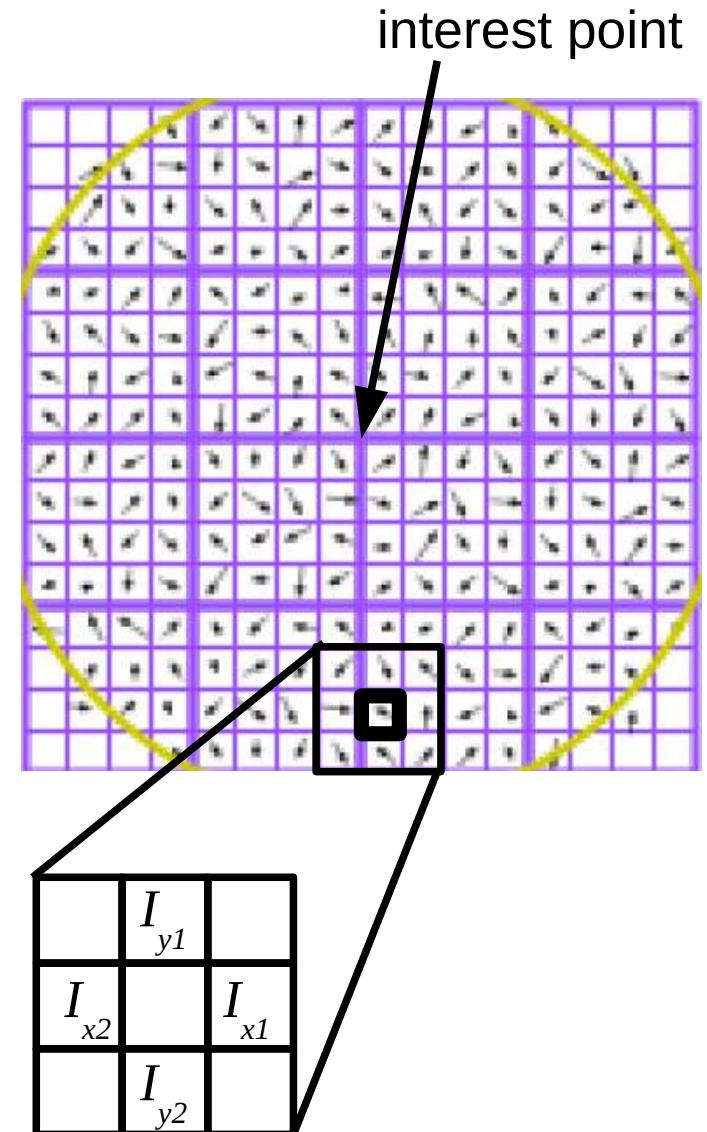
Step 1: Calculate the orientation and magnitude of the intensity gradient at all pixels surrounding the interest point.

This is done using the Gaussian smoothed image at the scale where the interest point was found.

Magnitude and orientation approximated using pixel differences.

$$mag = \sqrt{(I_{x1} - I_{x2})^2 + (I_{y1} - I_{y2})^2}$$

$$ori = \tan^{-1} \left(\frac{I_{y1} - I_{y2}}{I_{x1} - I_{x2}} \right)$$



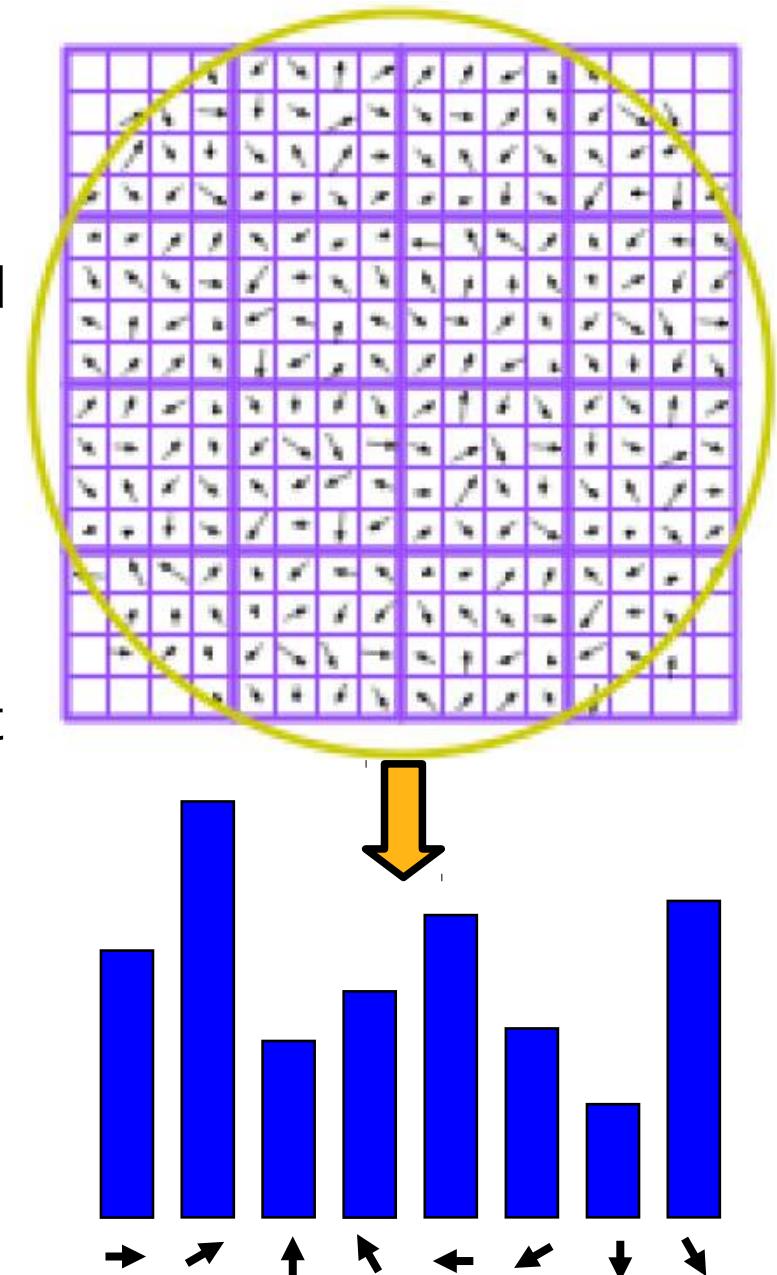
SIFT: feature descriptor

Step 2: Create a histogram of all the orientations around the interest point.

Each sample added to the histogram is weighted by its gradient magnitude and a Gaussian centred on the interest point.

Find dominant orientation, by finding peak in histogram.

Rotate all orientations so that dominant orientation points up.



SIFT: feature descriptor

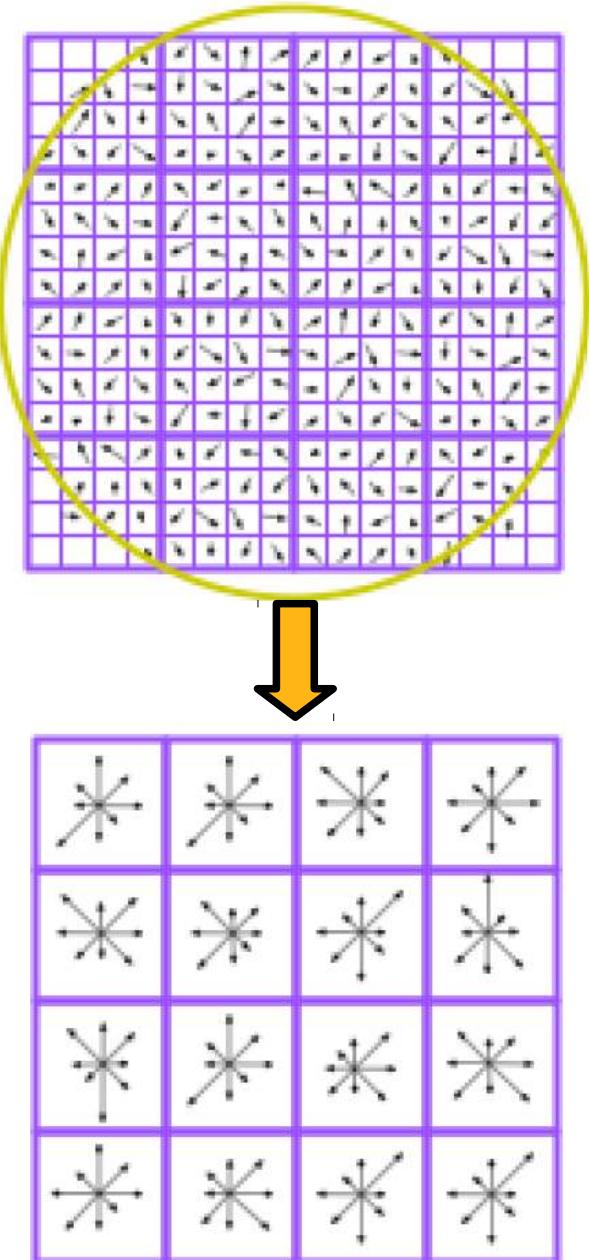
Step 3: Create separate histograms of all the orientations in 4x4 sub-windows around the interest point.

Each sample added to each histogram is weighted by its gradient magnitude and a Gaussian centred on the interest point.

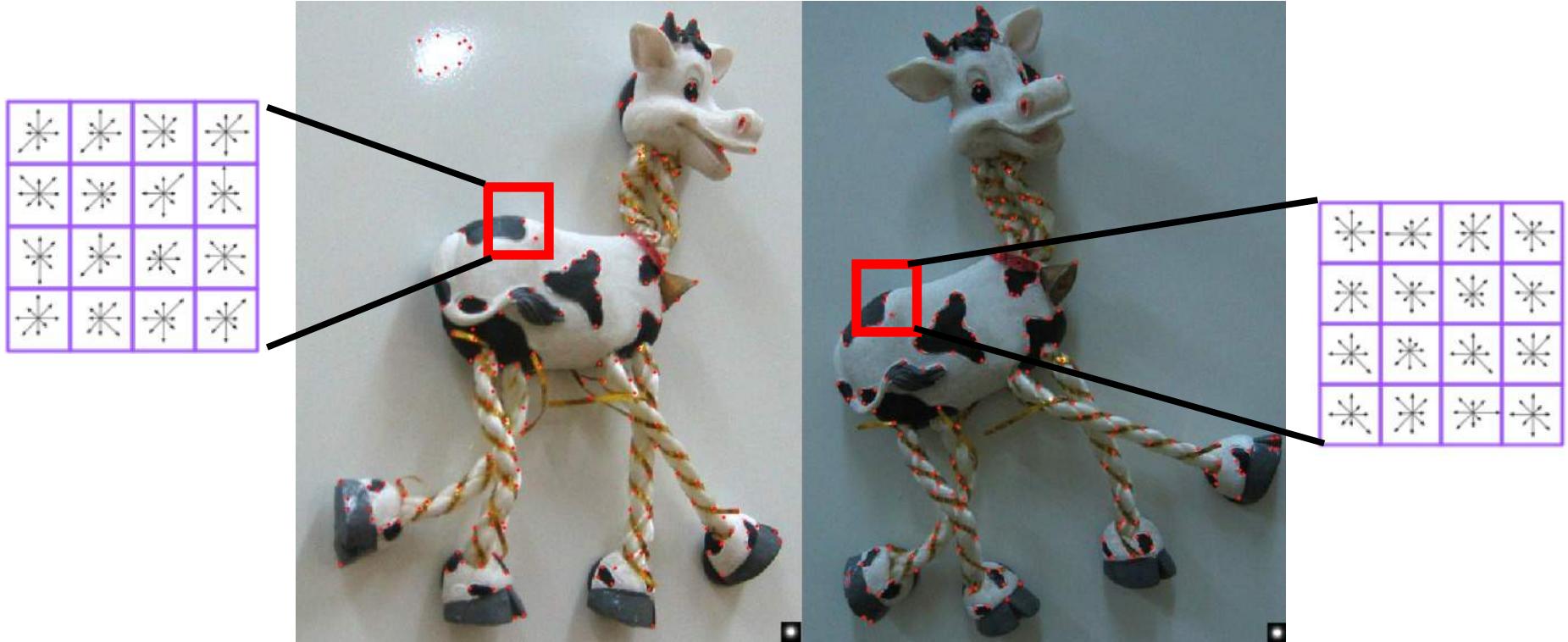
Use 8 orientation bins in each histogram.

The descriptor for each interest point is therefore a $4 \times 4 \times 8 = 128$ element vector.

This vector is normalised to unit length.



SIFT: feature descriptor and matching



Descriptor: 128 element vector of intensity gradient orientations around the interest point.

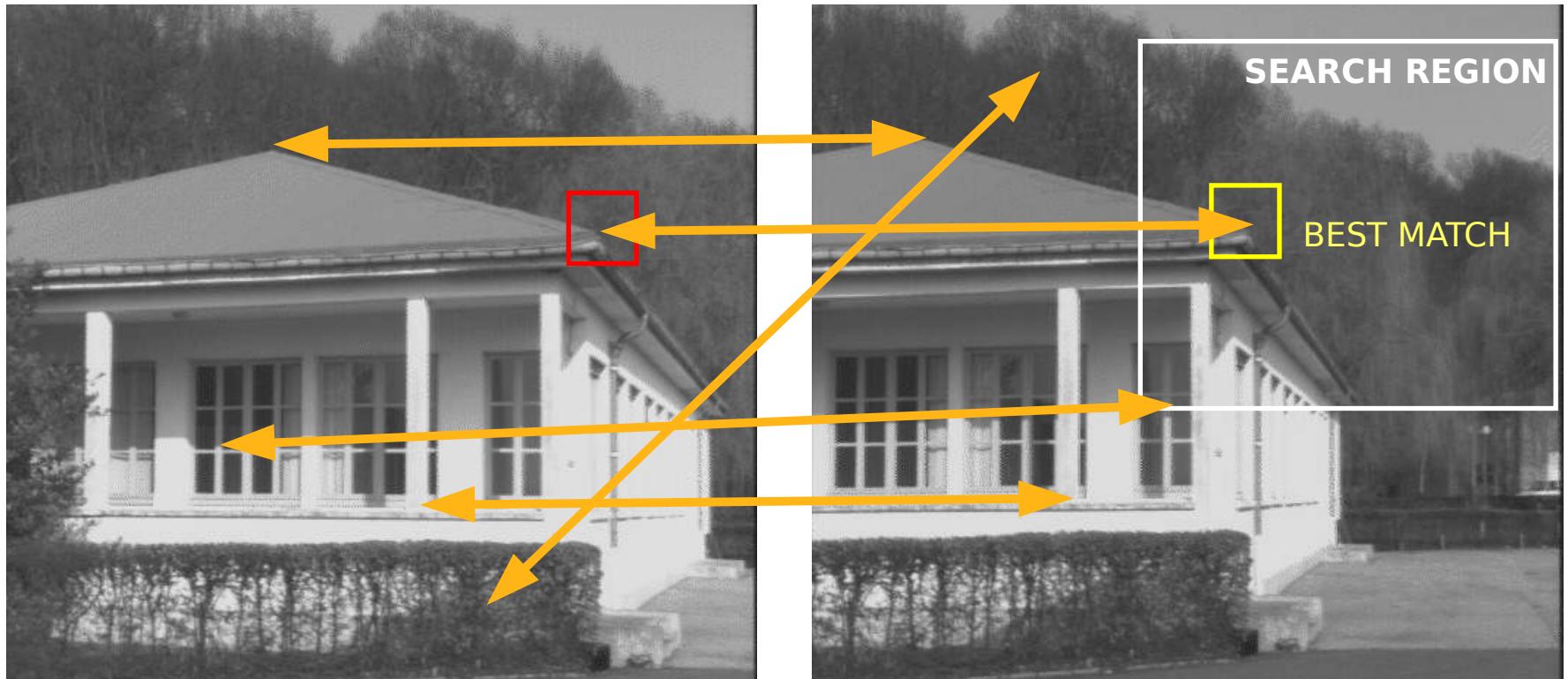
Similarity measure: Euclidean distance between vectors.

Robust to translation, rotation, scale, changes in viewpoint and illumination.

Matching: dealing with outliers

Whether a correlation-based method or a feature-based method is used, search is required to find points that are most similar.

These “most similar” points are *putative matches*

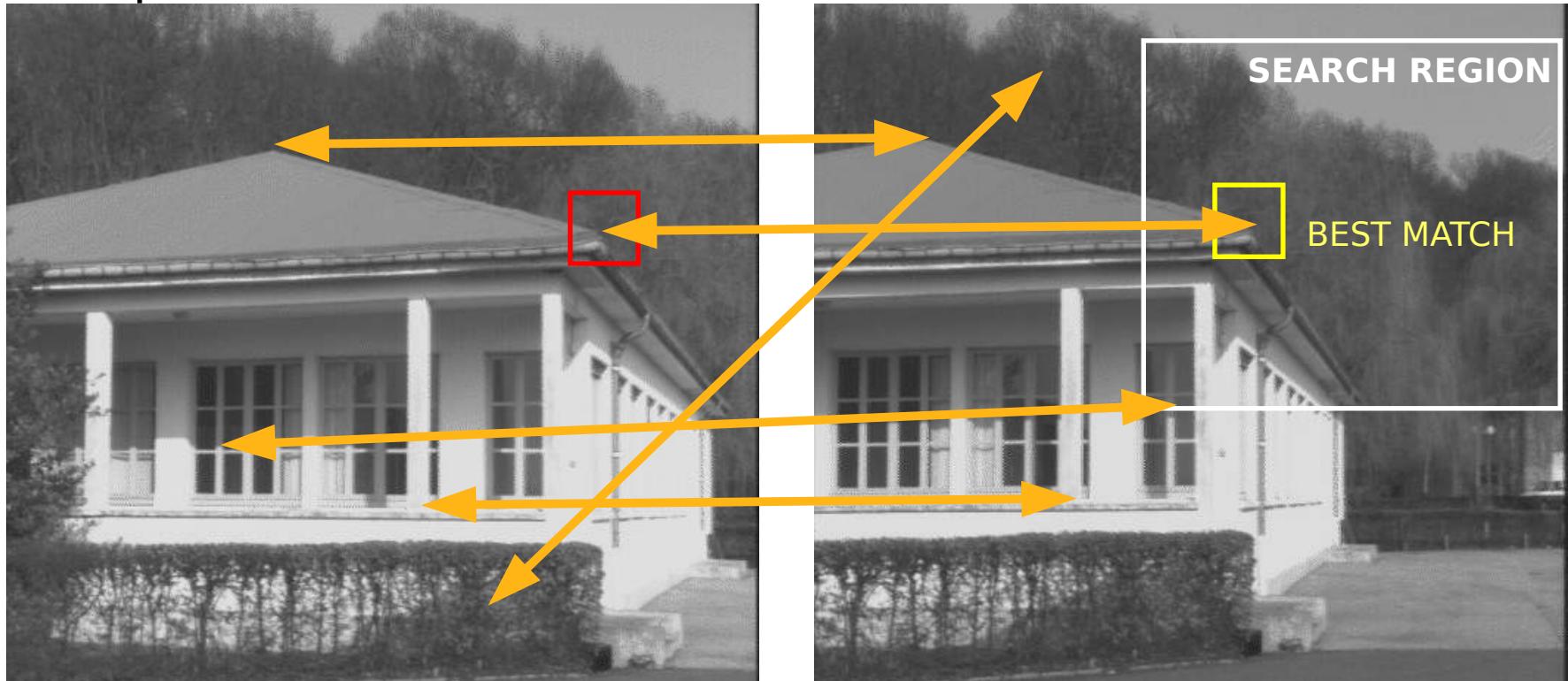


Some of these putative matches may be correct (“**inliers**”) but others may be wrong (“**outliers**”).

How do we find the true correspondence between images despite these matching errors?

Matching: dealing with outliers

Need to estimate transformation between images despite erroneous correspondences.



1. (Extract features – if using feature-based method)
 2. Compute putative matches
 3. Find most likely transformation (i.e. the one with the most inliers and fewest outliers)
- use the **RANSAC** (= RANdom SAMpling & Consensus) algorithm

RANSAC: algorithm

Objective:

Robust fit of model to data set which contains outliers

Requirements:

1. Data consists of inliers and outliers
2. A parameterized model explains the inliers

Procedure:

1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. Fit the model to this subset of data
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction)
4. Count the number of inliers (the *consensus* set). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

After N trials select the model parameters with the highest support and re-estimate the model using all the points in this subset.

RANSAC: simple correspondence example

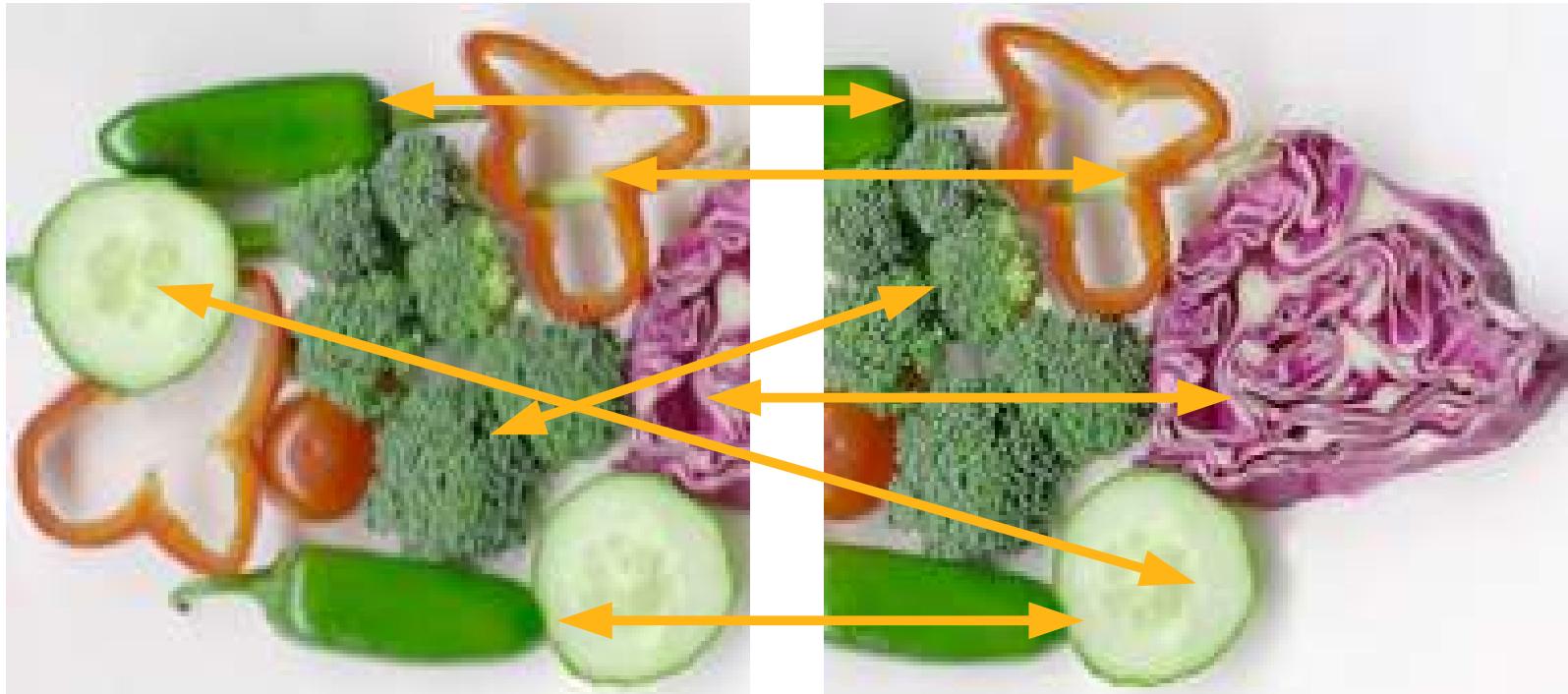
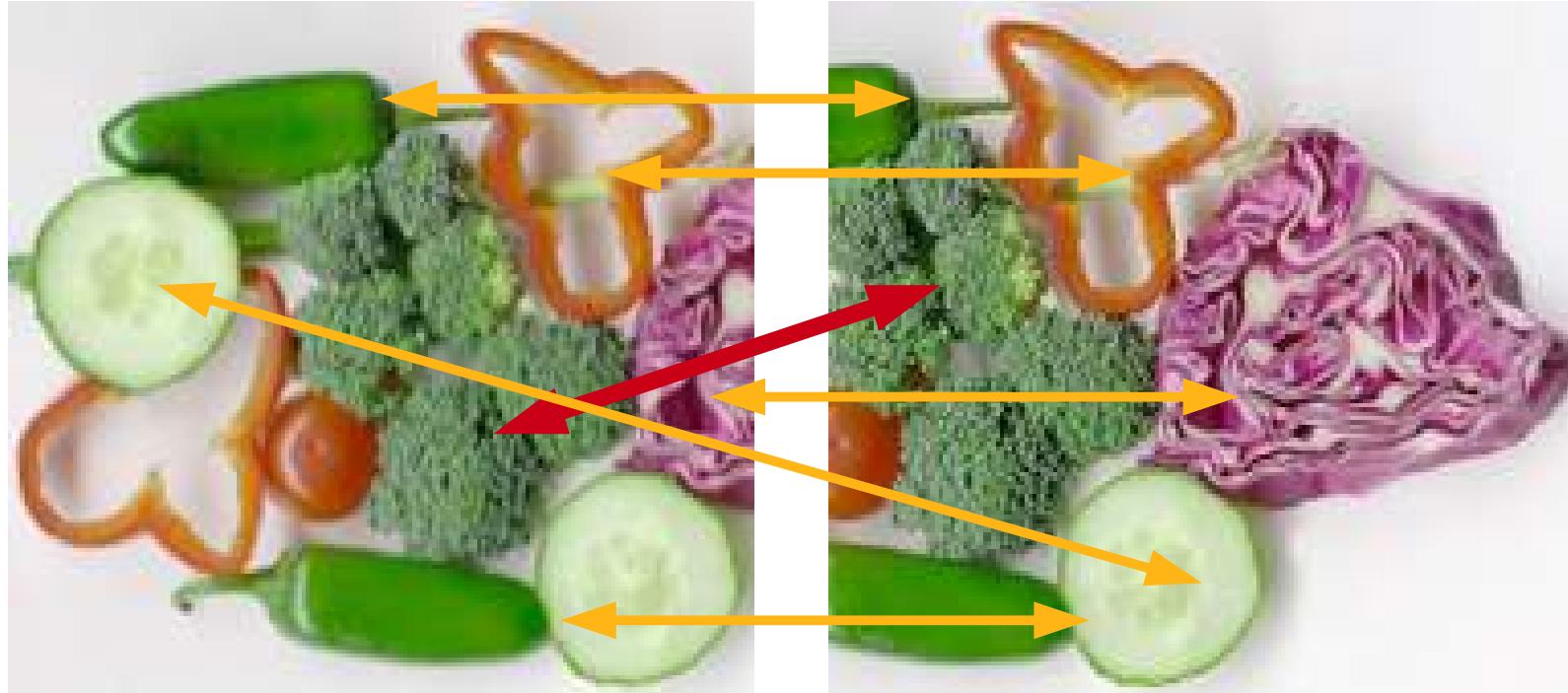


Image shows a set of putative matches between points in two images

Assume the two images are related by a pure translation.
i.e. the model we wish to fit is a translation by Δx and Δy .

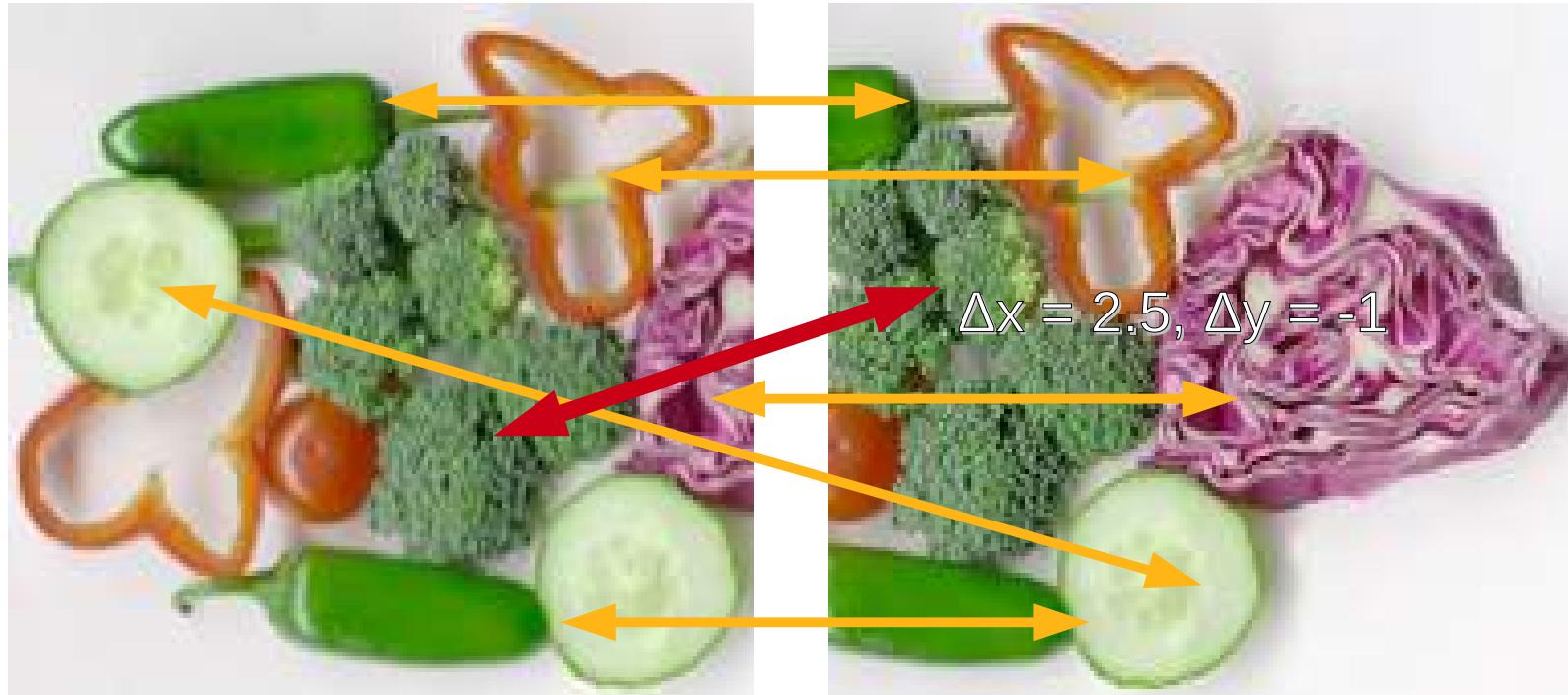
One putative match is sufficient to define Δx and Δy .

RANSAC: simple correspondence example



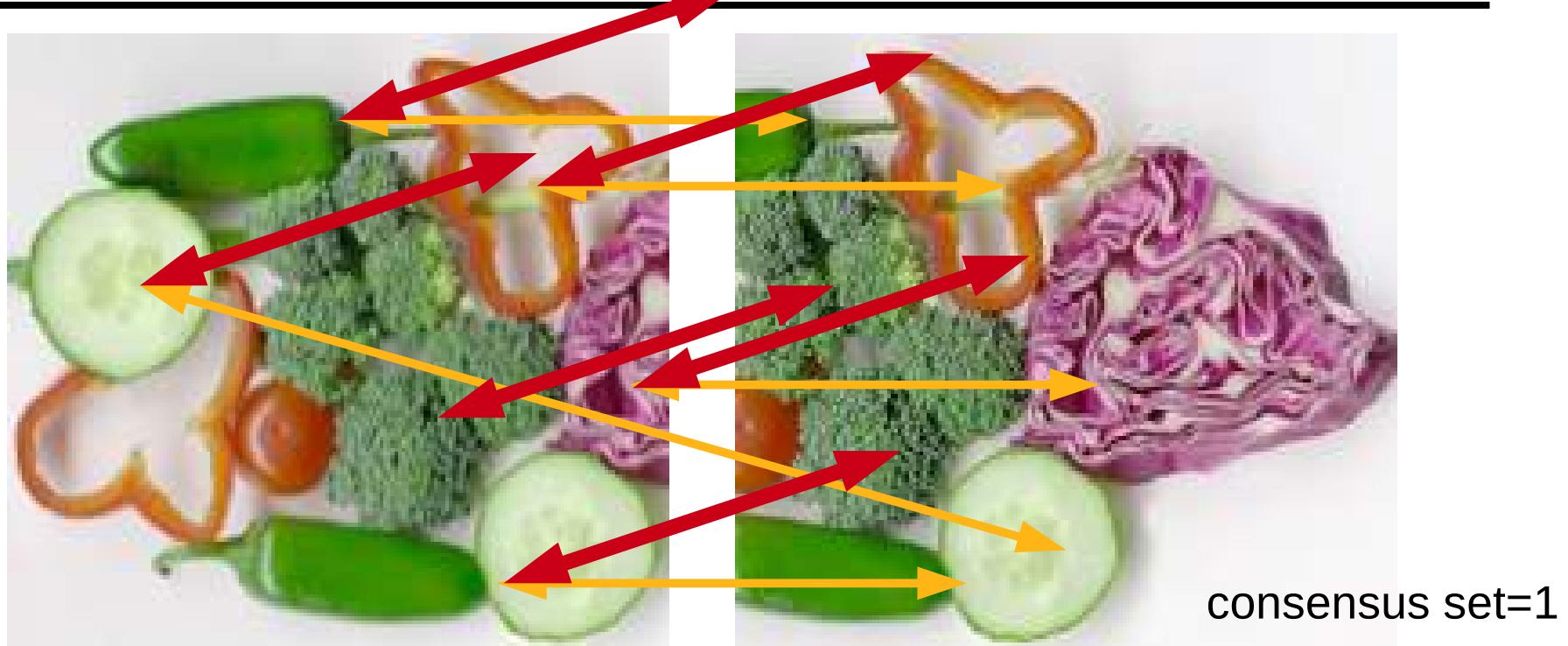
1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. Fit the model to this subset of data
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus* set). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: simple correspondence example



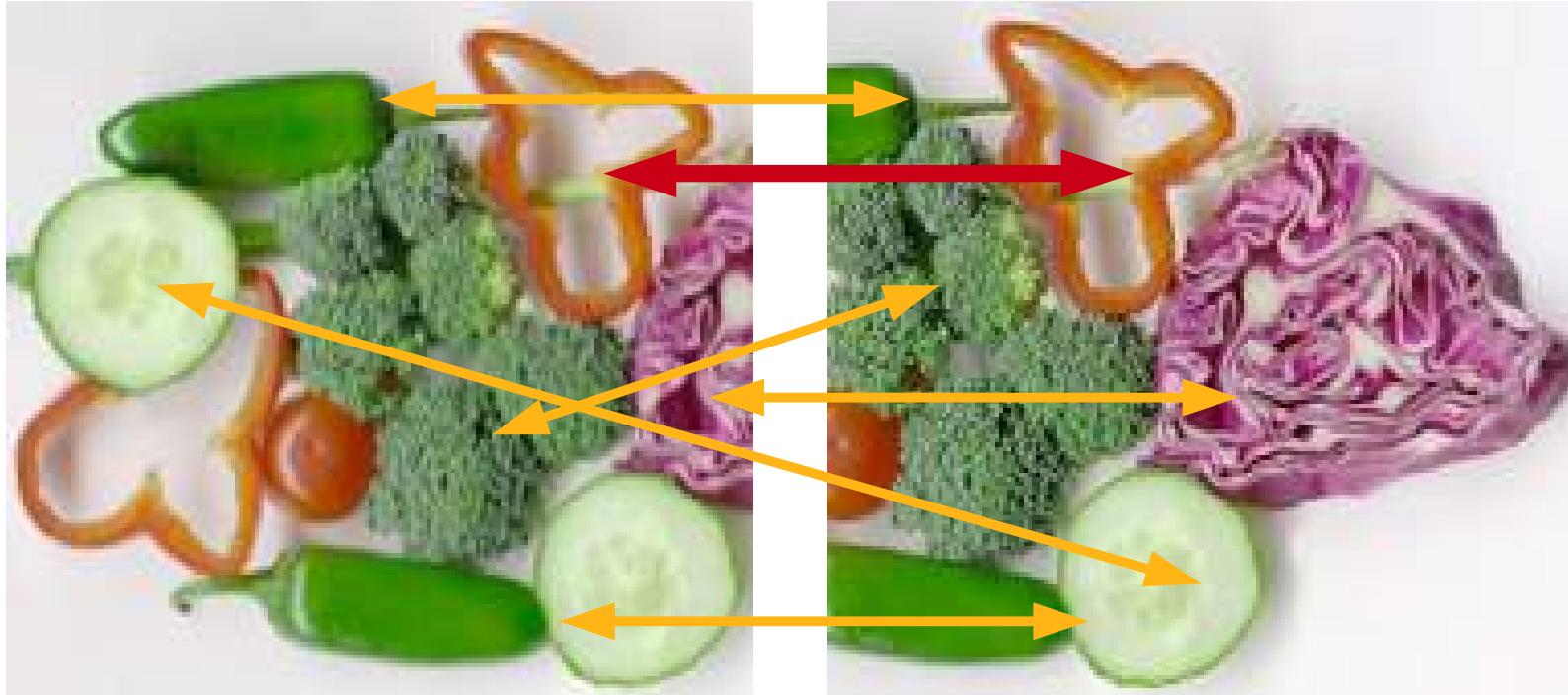
1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. **Fit the model to this subset of data**
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus* set). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: simple correspondence example



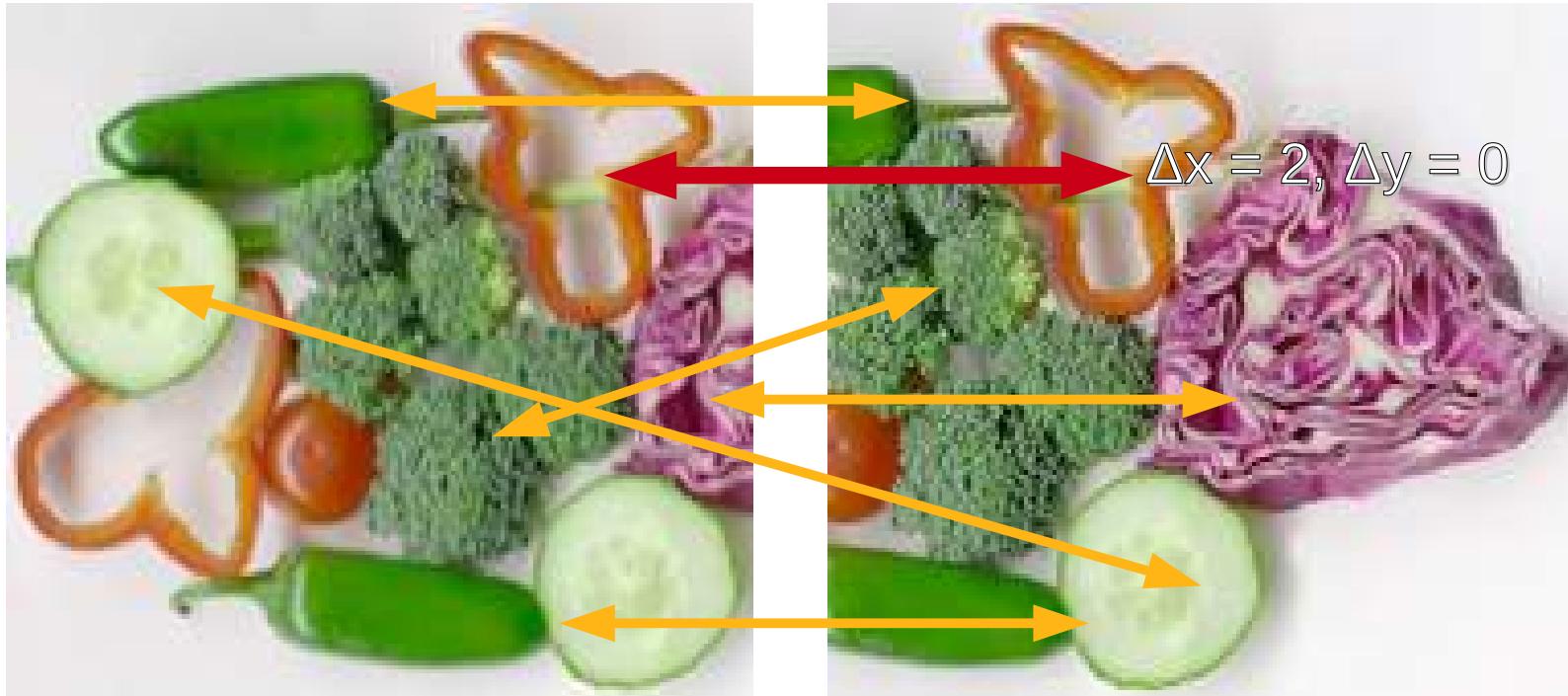
1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. Fit the model to this subset of data
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus set*). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: simple correspondence example



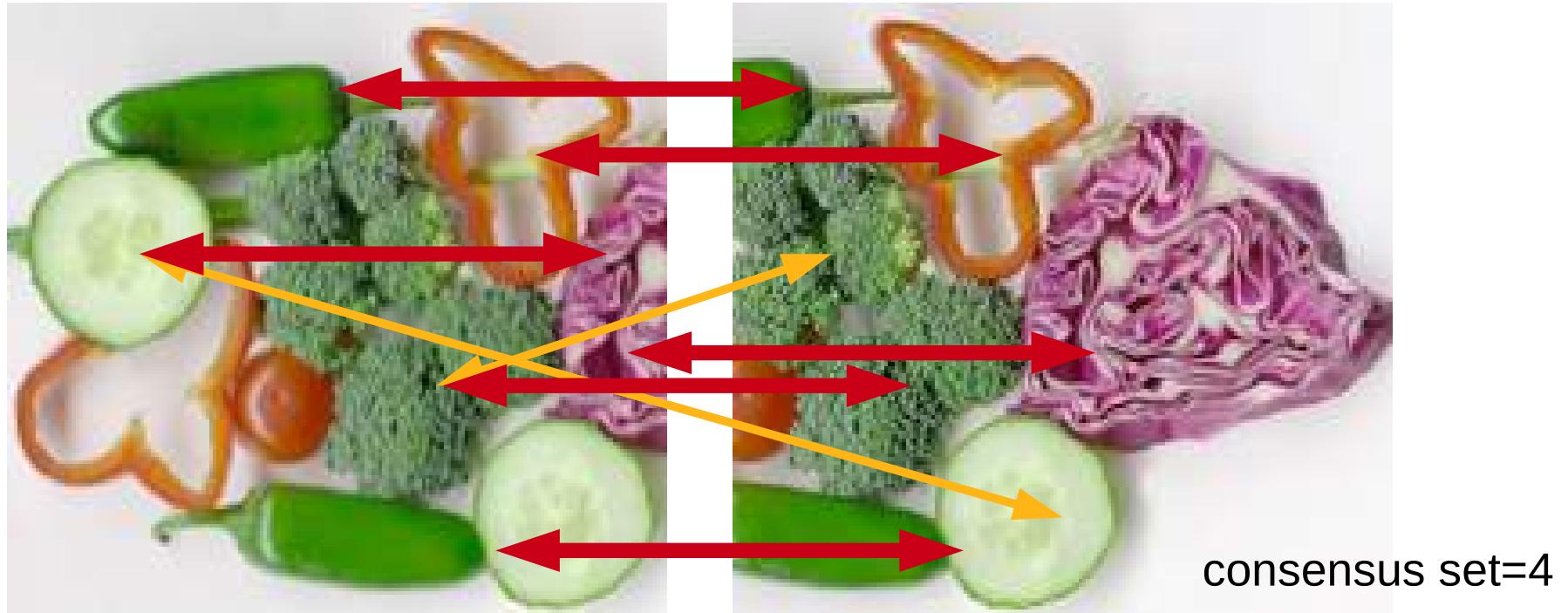
1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. Fit the model to this subset of data
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus* set). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: simple correspondence example



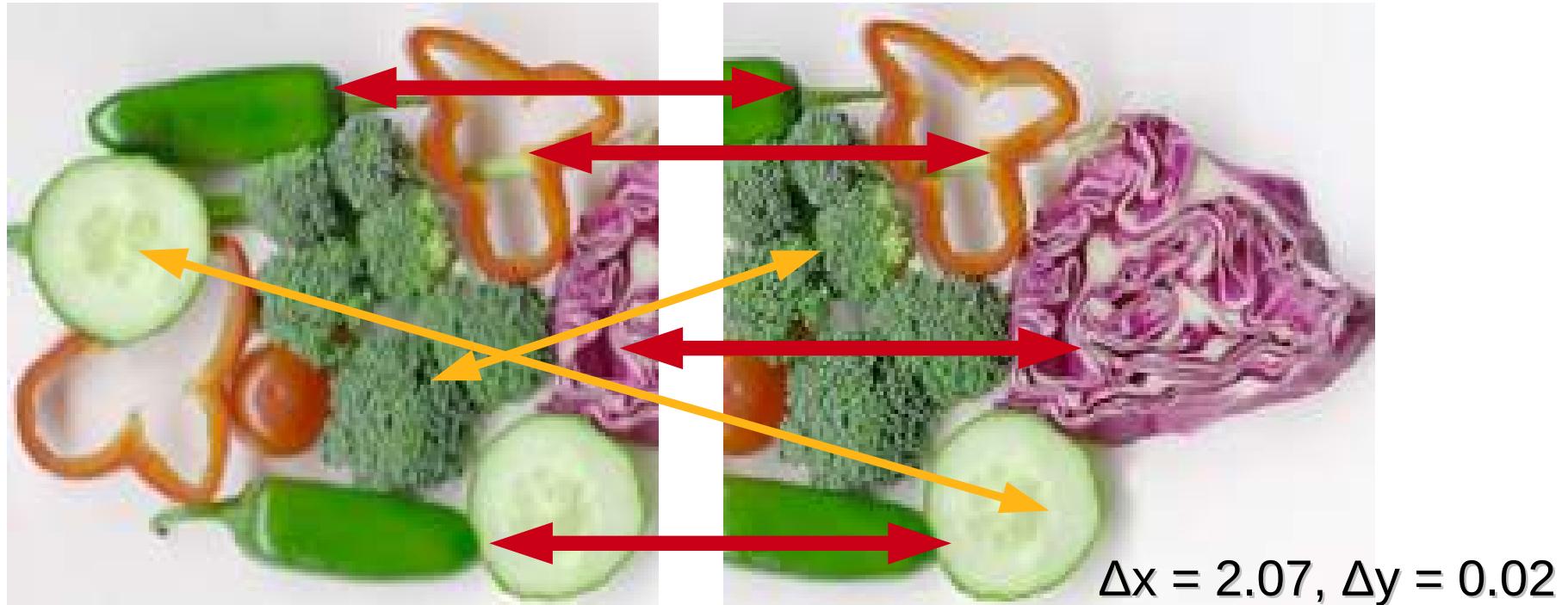
1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. **Fit the model to this subset of data**
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus* set). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: simple correspondence example



1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. Fit the model to this subset of data
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus set*). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: simple correspondence example



After N trials select the model parameters with the highest support and re-estimate the model using all the points in this subset.

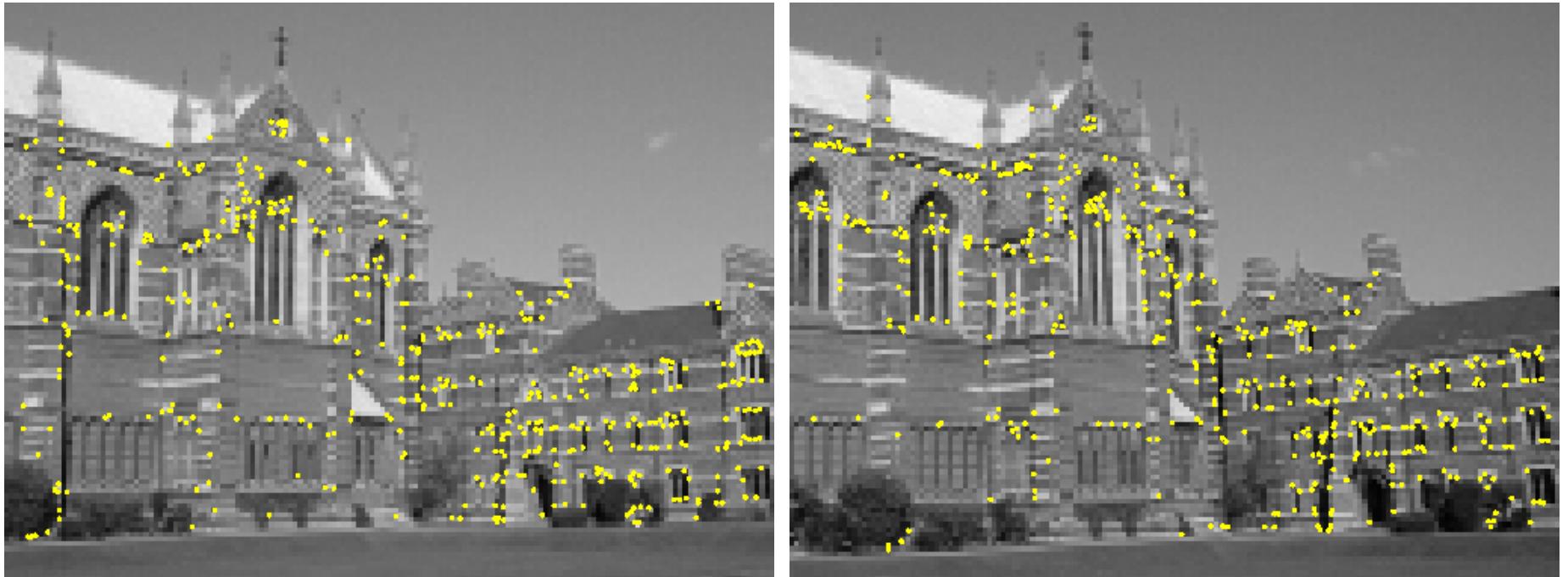
RANSAC: real correspondence example

Generally, the correspondence between views will be more complex than a pure translation.

Translation and rotation of the camera results in more complex transformations between images.

We can still estimate the parameters of this transformation by sampling more pairs of points (e.g. 4 pairs of putative matches)

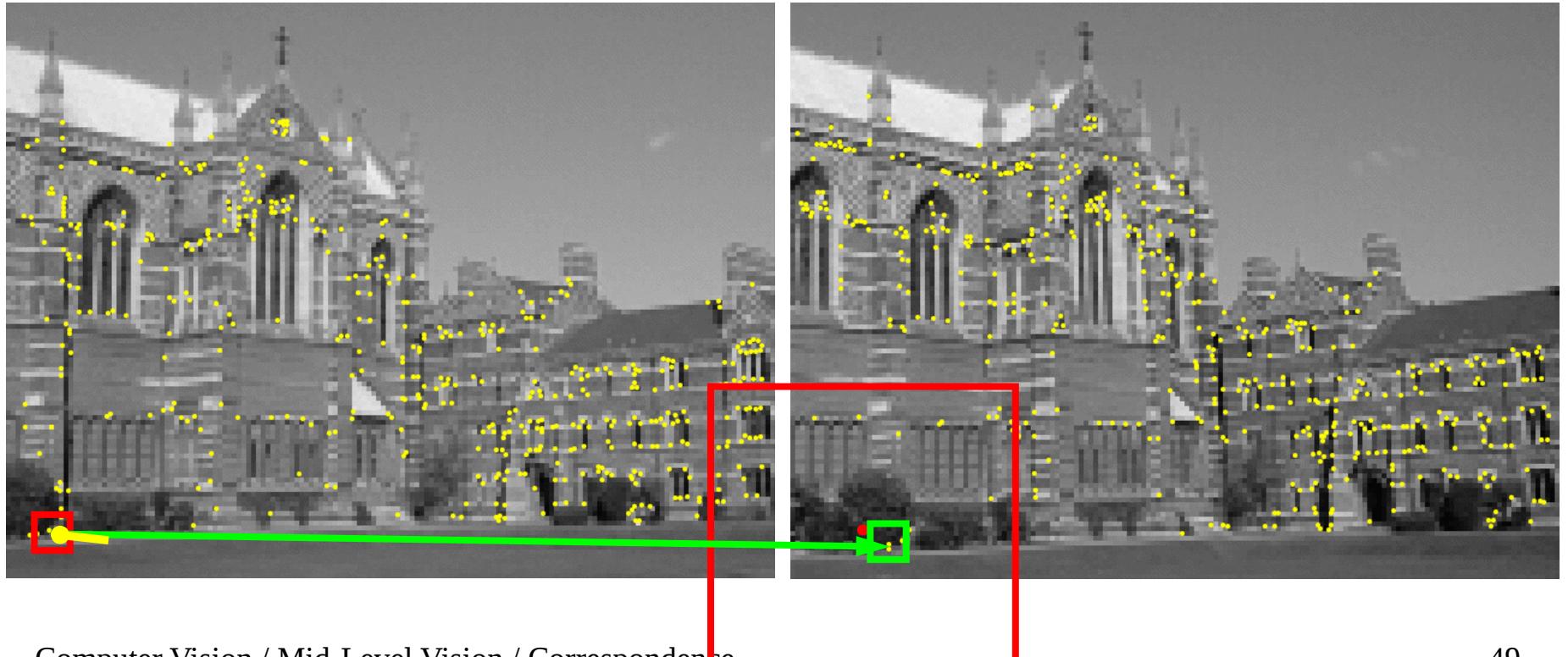
In this example approx 500 interest points have been extracted with the Harris corner detector



RANSAC: real correspondence example

For each interest point the best match has been found within a square search window (here 300 pixels) using SSD

These putative matches are shown using a line pointing from the interest point in the left image to the pixel location of the corresponding point in the right image



RANSAC: real correspondence example

This results in 188 initial matches (which exceed some similarity threshold)



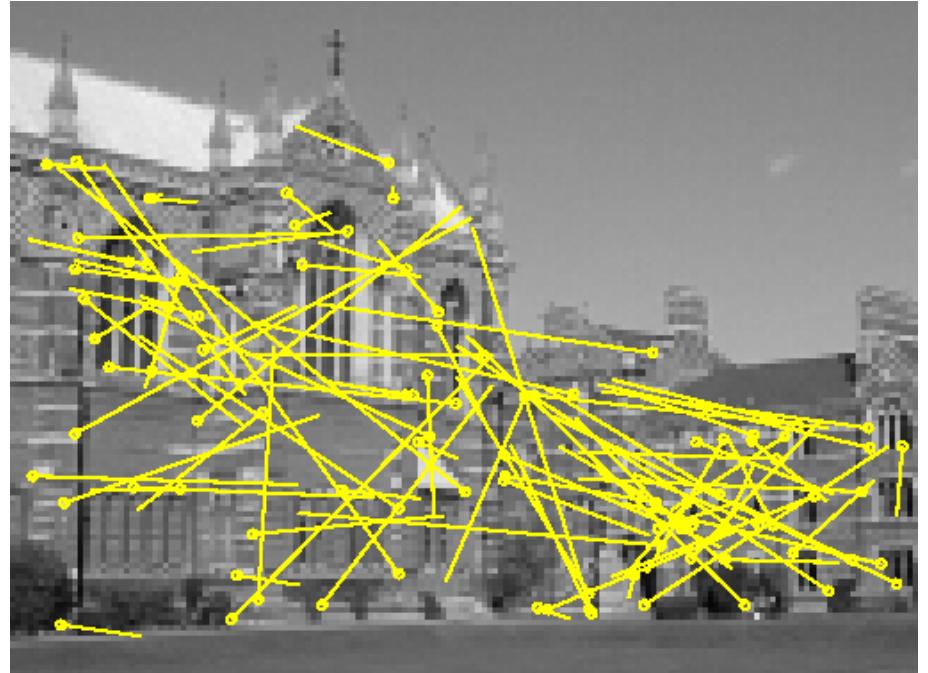
RANSAC: real correspondence example

Applying RANSAC to determine the transformation between the camera locations, results in a model that is consistent with 99 matches and inconsistent with 89 matches.

Note, RANSAC allows correspondence to be found even in the presence of many outliers.



99 inliers



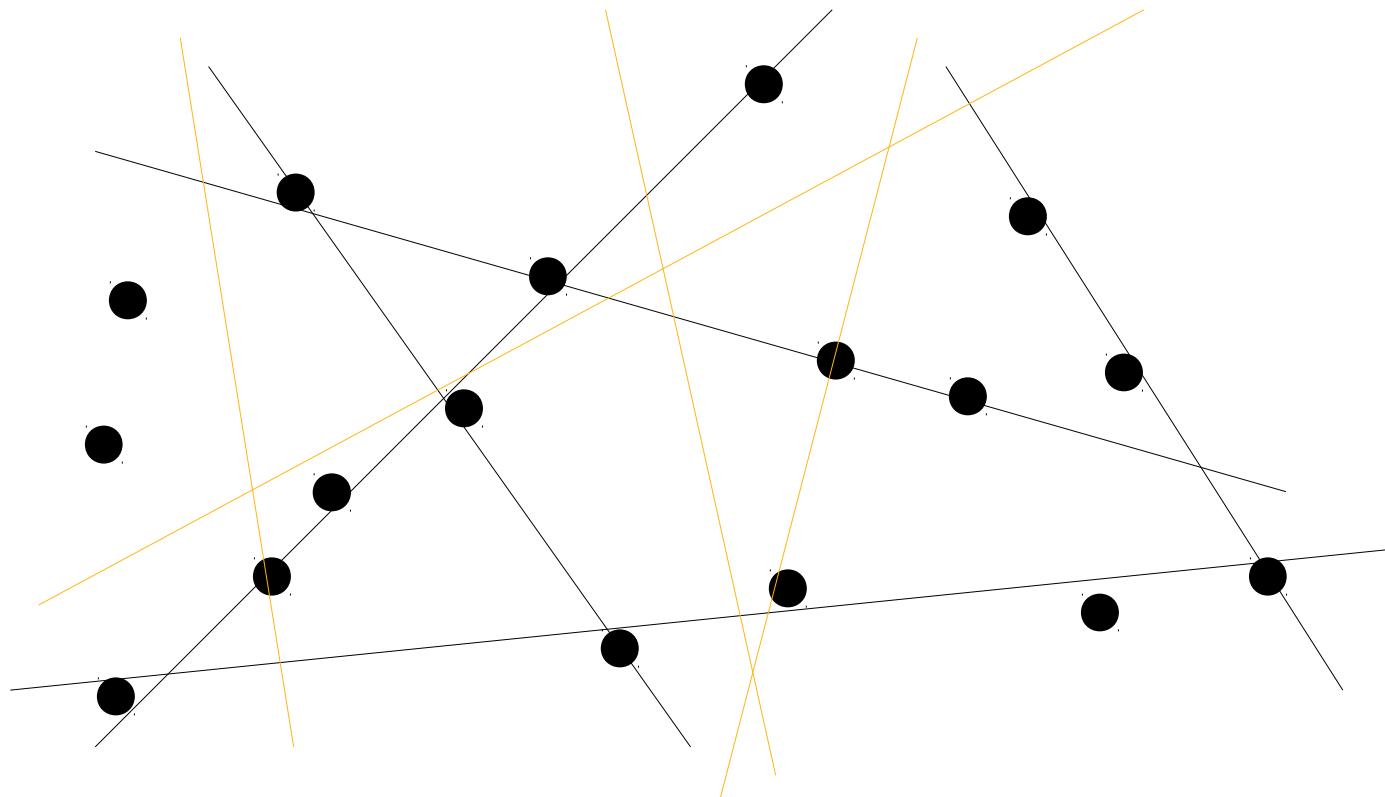
89 outliers

RANSAC for fitting

Recall, fitting algorithms (used for segmentation):

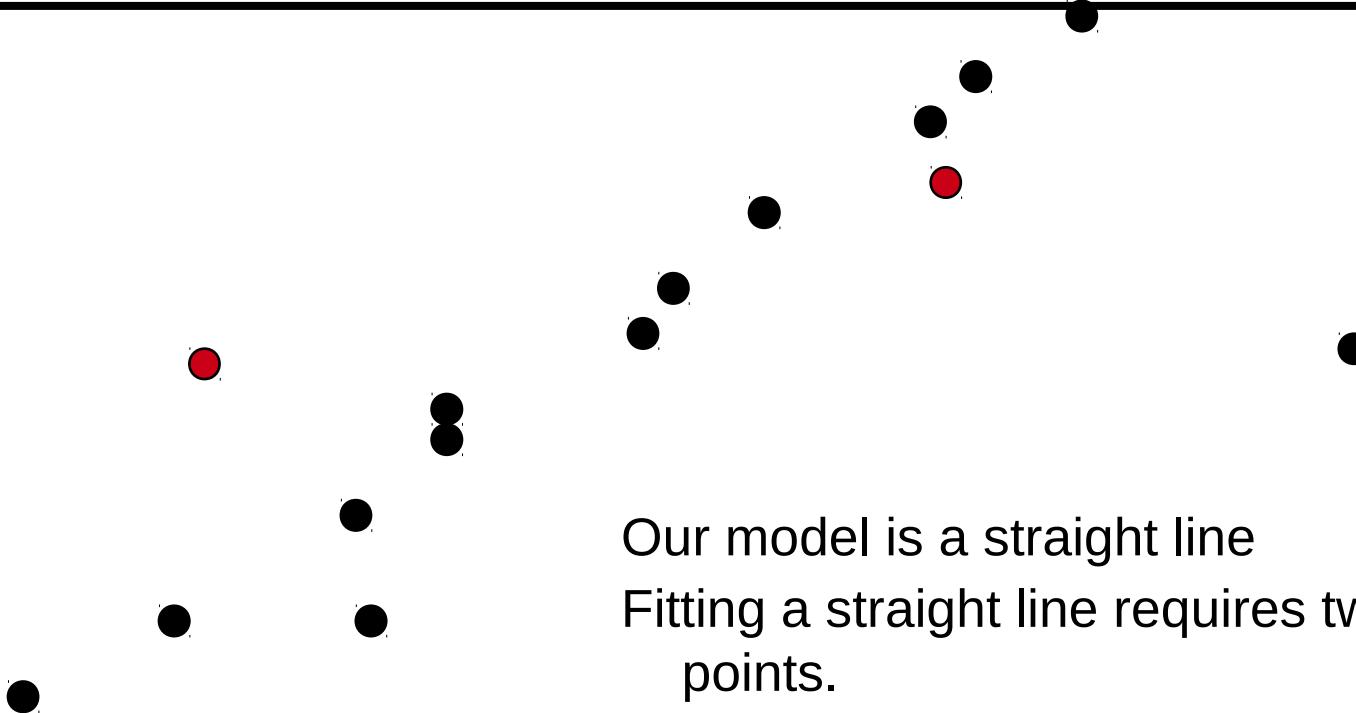
A class of methods that try to use a mathematical model to represent a set of tokens.

e.g. to fit a straight line to a set of points



One algorithm for fitting a model to data is the Hough Transform
RANSAC can also be used

RANSAC: line fitting example

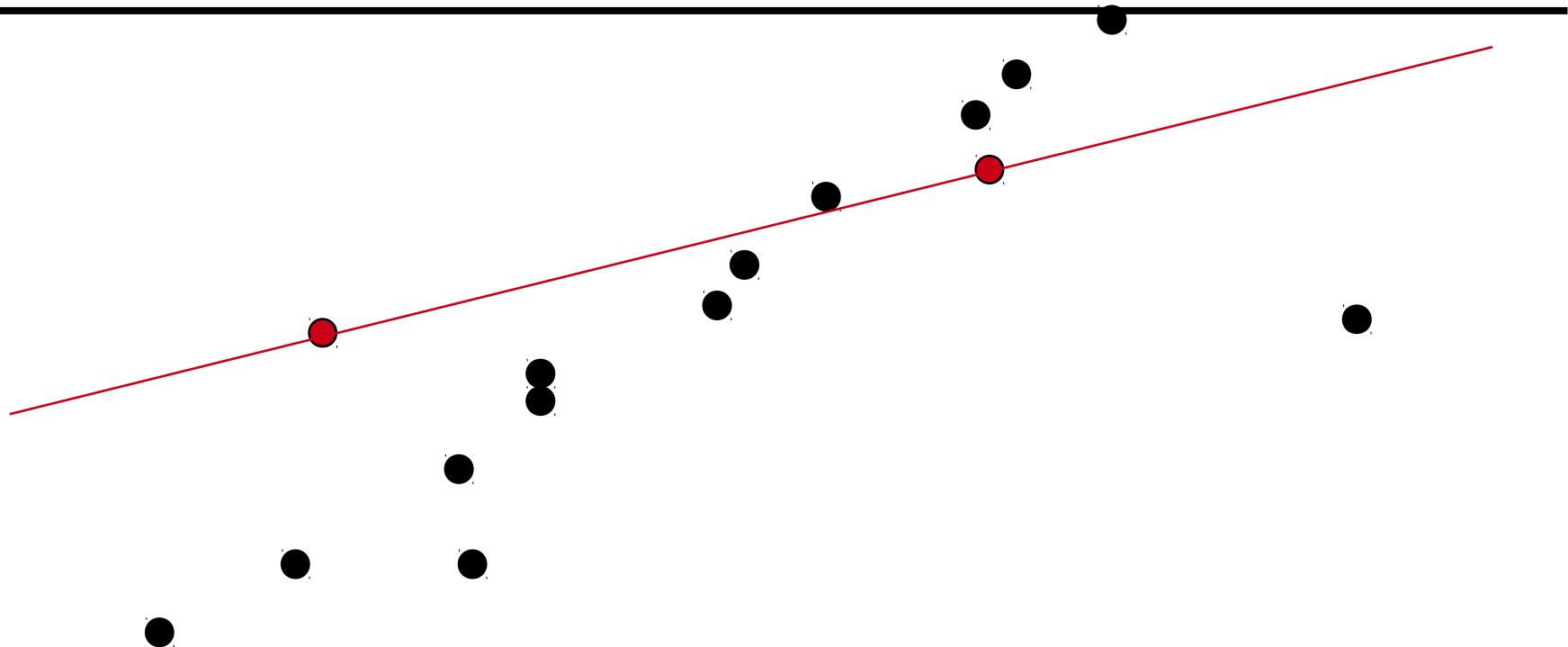


Our model is a straight line
Fitting a straight line requires two
points.

Hence, our sample size is two.

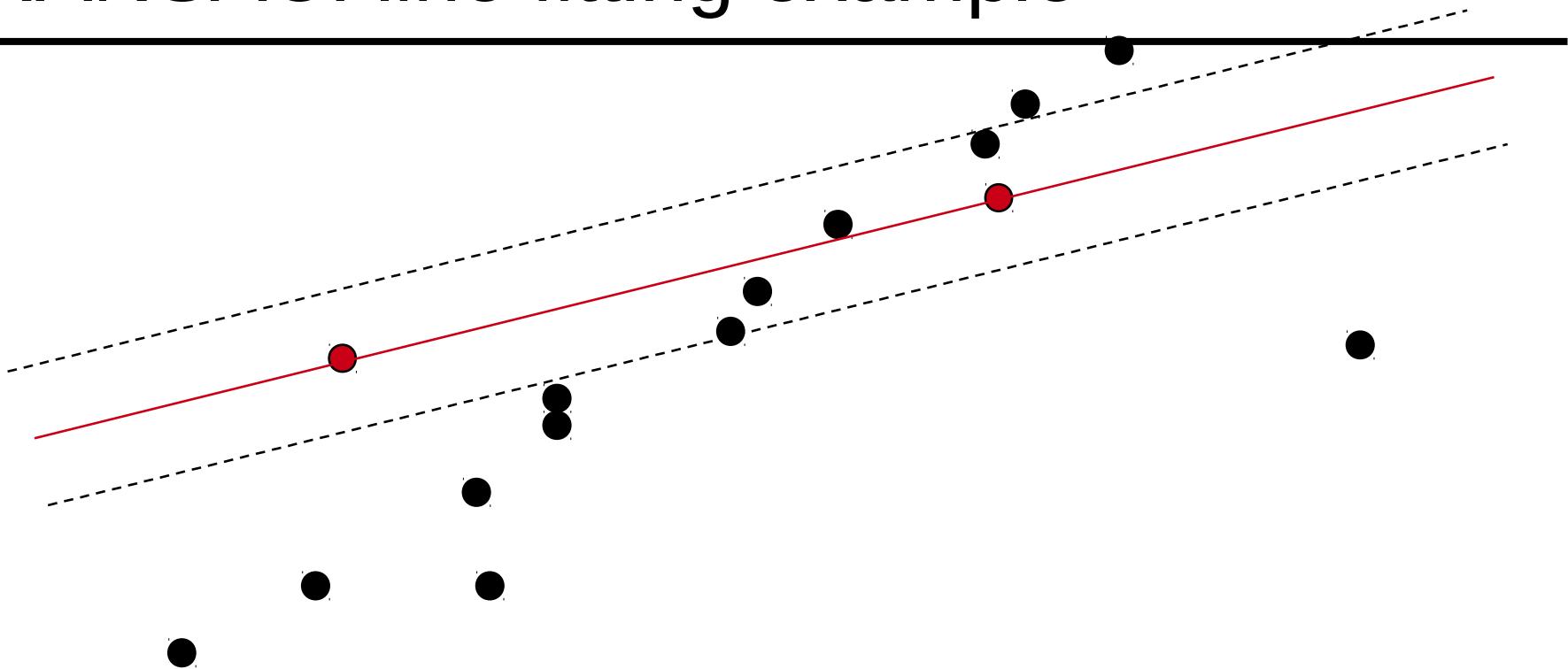
1. Randomly choose a minimal subset (*a sample*) of data points necessary to fit the model
2. Fit the model to this subset of data
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (*the consensus set*). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: line fitting example



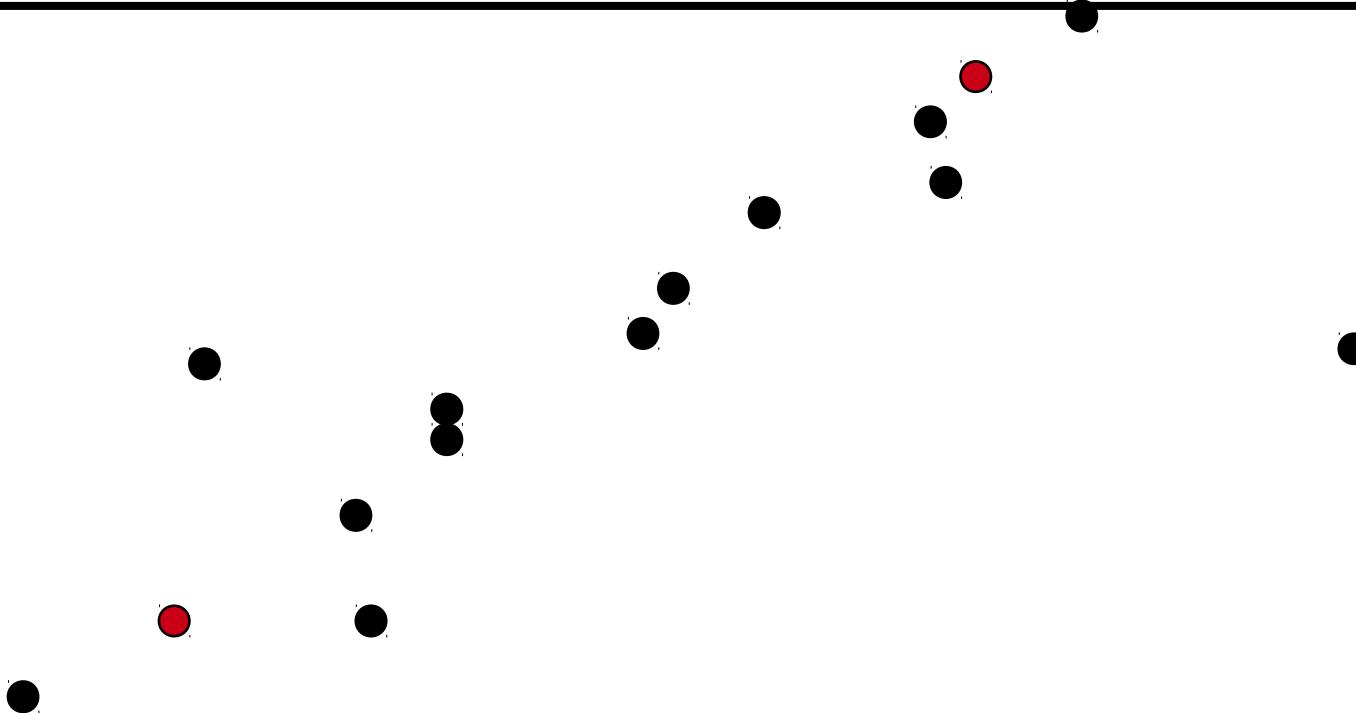
1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. **Fit the model to this subset of data**
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus* set). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: line fitting example



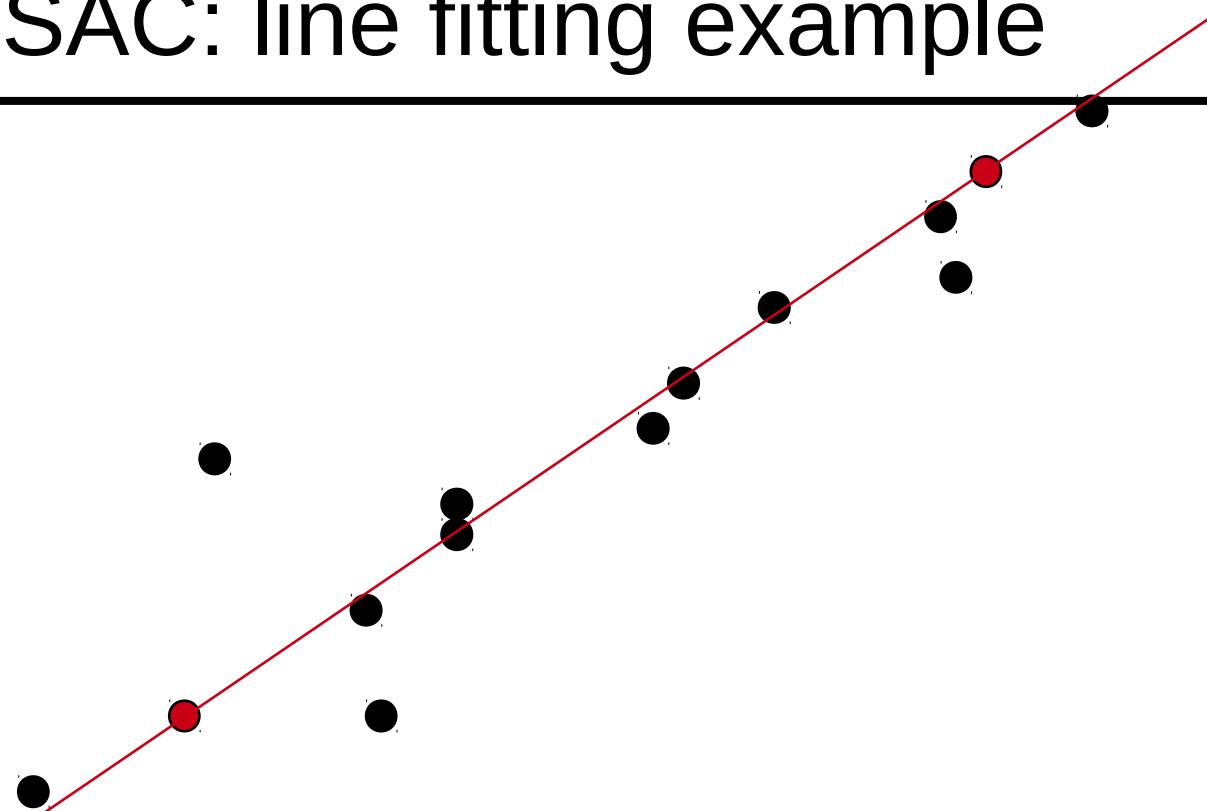
1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. Fit the model to this subset of data
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus* set). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: line fitting example



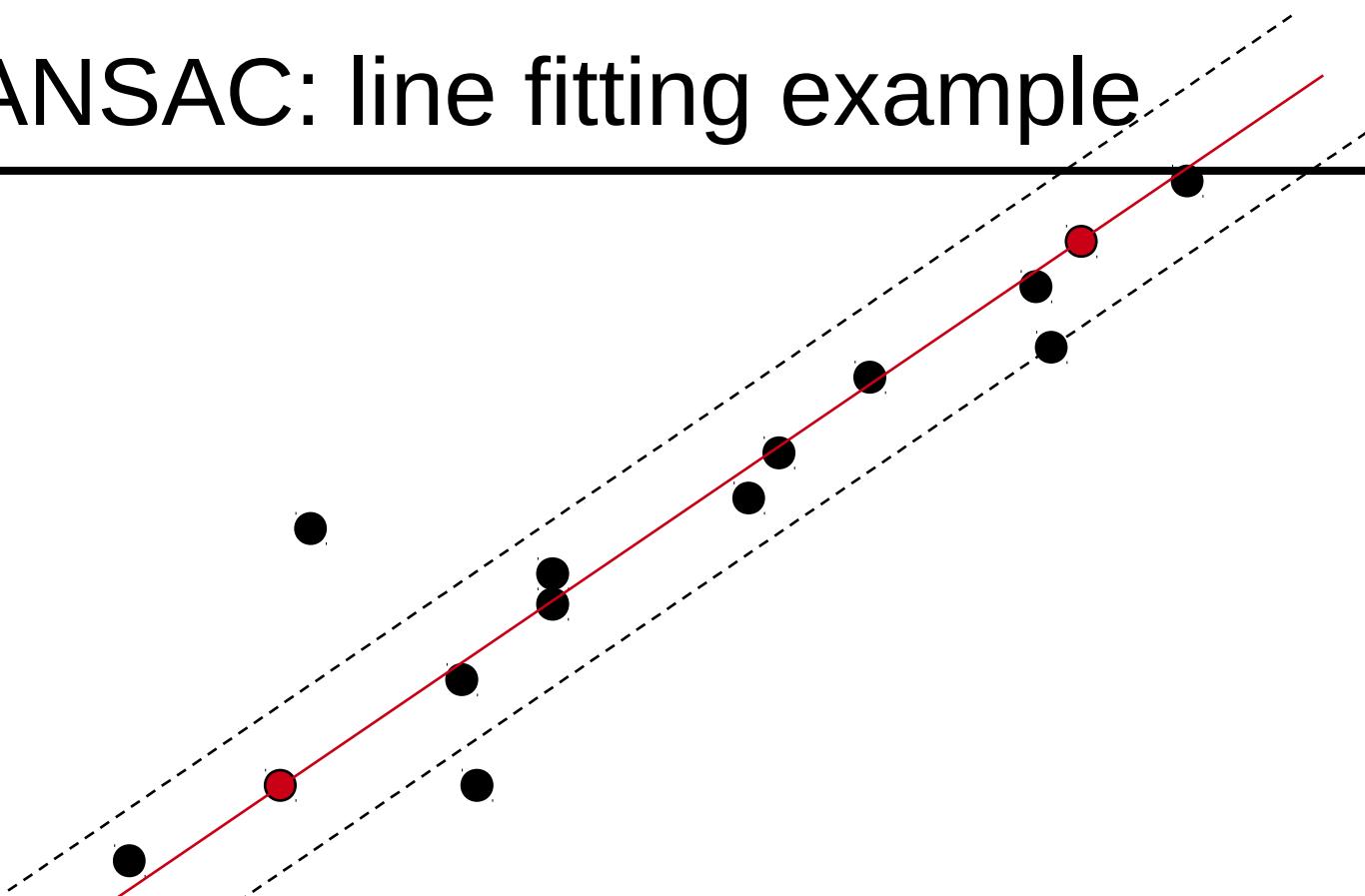
1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. Fit the model to this subset of data
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus* set). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: line fitting example



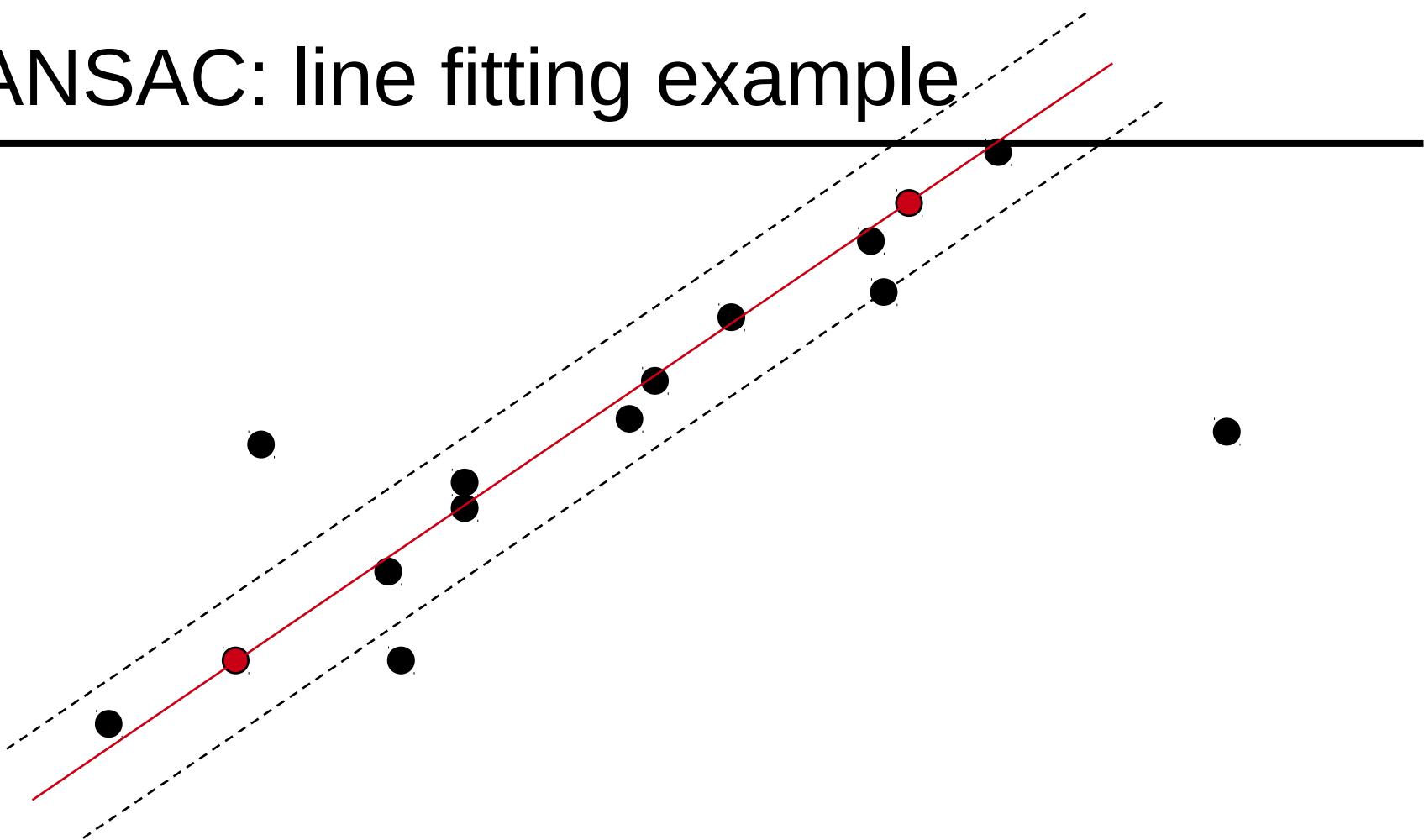
1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. **Fit the model to this subset of data**
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus* set). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: line fitting example



1. Randomly choose a minimal subset (a *sample*) of data points necessary to fit the model
2. Fit the model to this subset of data
3. Test all the other data points to determine if they are consistent with the fitted model (i.e. if they lie within a distance t of the model's prediction).
4. Count the number of inliers (the *consensus* set). Size of consensus set is model's *support*
5. Repeat from step 1 for N trials

RANSAC: line fitting example



After N trials select the model parameters with the highest support and re-estimate the model using all the points in this subset.

RANSAC: pros and cons

Advantages:

- Simple and effective
- General method suited for a wide range of model fitting problems
 - » e.g. for segmentation by model fitting
 - » e.g. for finding camera transformation given stereo views
 - » e.g. for finding object trajectory given video

Disadvantages:

- Sometimes very many iterations are required if percentage of outliers is high.
- Lots of parameters to tune

Summary

Correspondence Problem

Finding matching image elements across images

- general problem arising in:
 - stereo (multiple cameras)
 - video (multiple times)
 - object recognition (comparing images)
- similar to grouping:
 - grouping is looking for similar elements in a single image
 - correspondence is looking for the same elements in multiple images

Summary

Solving the Correspondence Problem

1. Which image locations to match
 - a. all locations (correlation-based method)
 - b. selected interest points (feature-based methods: Harris, SIFT)
2. What properties to match
 - a. image intensities
 - b. a descriptor of image properties (SIFT)
3. Where to look for matches
 - a. exhaustive search across entire image
 - b. restricted search (constrained by task knowledge)
4. How to evaluate matches
 - a. similarity (correlation, normalised correlation, correlation coefficient)
 - b. differences (SSD, Euclidean distance, SAD)
5. How to find true correspondence (eliminate false matches)
RANSAC

Computer Vision (7CCSMCVI / 6CCS3COV)

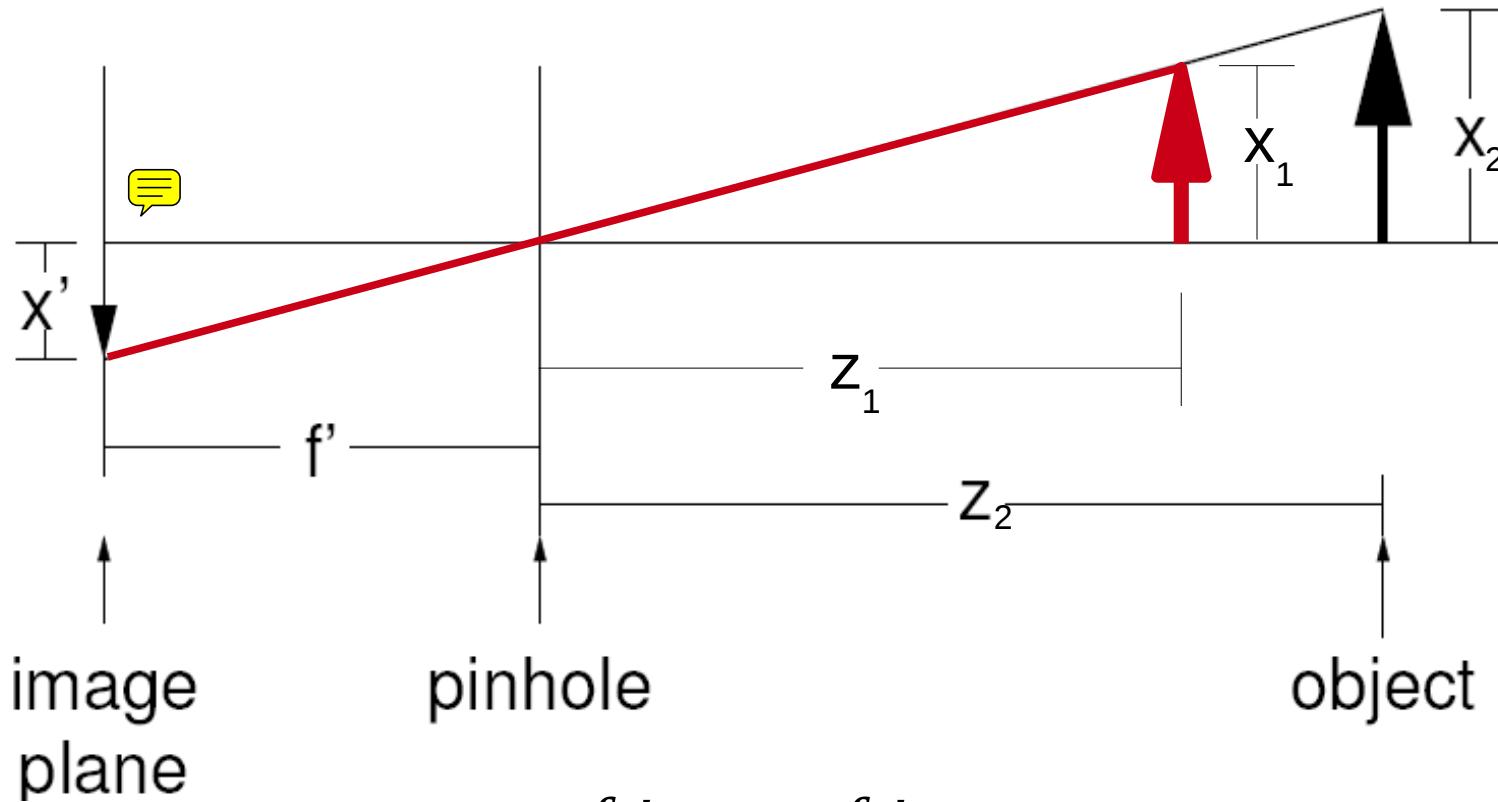
Recap

- **Image formation**
- **Low-level vision**
- **Mid-level vision**
 - grouping and segmentation (finding matching elements within an image)
 - correspondence problem (finding matching elements across images)
 - for all locations or selected interest points
 - comparing image intensities or descriptors
 - finding matches by search
 - determining similarity between elements
 - dealing with false matches by model fitting
 - Stereo and Depth ← Today
- **High-level vision**

Today

- Stereo vision
 - stereo camera geometry
 - » coplanar cameras (simple case)
 - » non-coplanar cameras (complex case)
 - disparity measurement
 - » calculating depth
 - correspondence
 - » stereo constraints used to solve the correspondence problem
- Other cues to depth
 - Binocular
 - Oculomotor
 - Monocular
 - Motion

Why is stereo vision important?

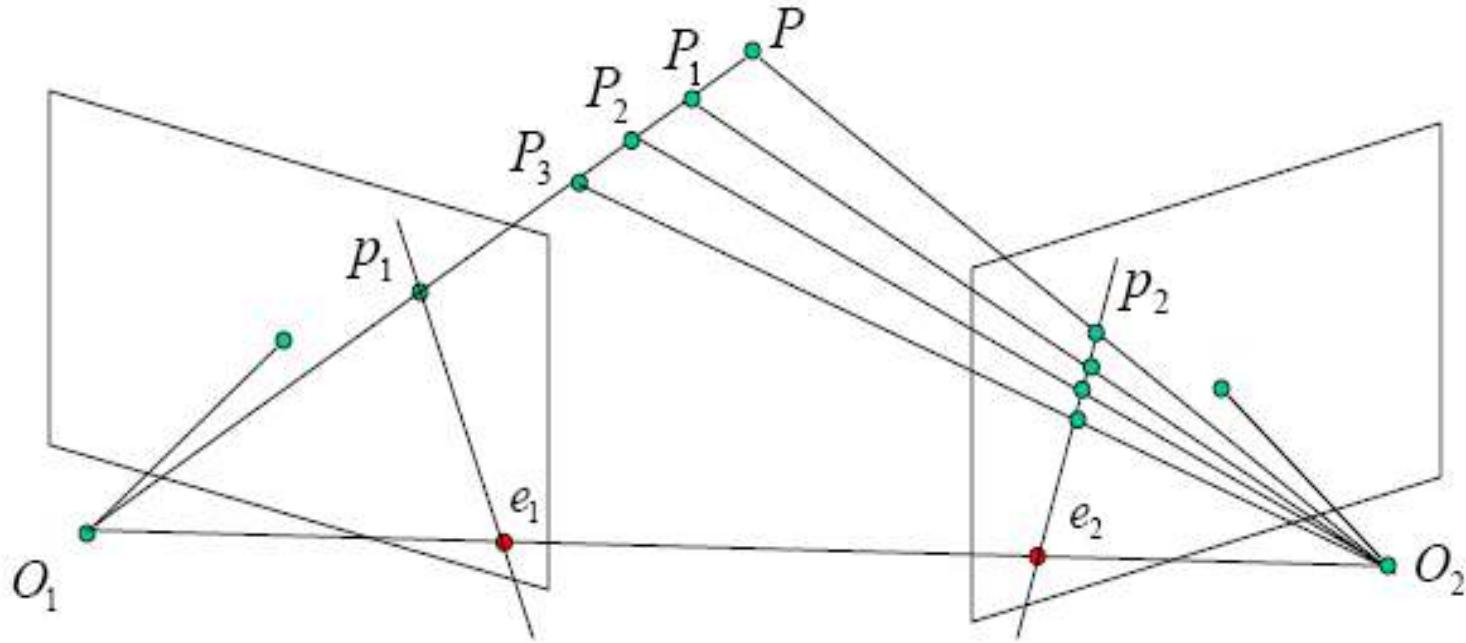


$$x' = \frac{f'}{Z_1} X_1 = \frac{f'}{Z_2} X_2$$

A camera projects 3D points onto a 2D plane

3D points on the same line-of-sight have the same 2D image location (i.e. imaging results in depth information loss)

Why is stereo vision important?



Depth information can be recovered using two images and a knowledge of geometry.

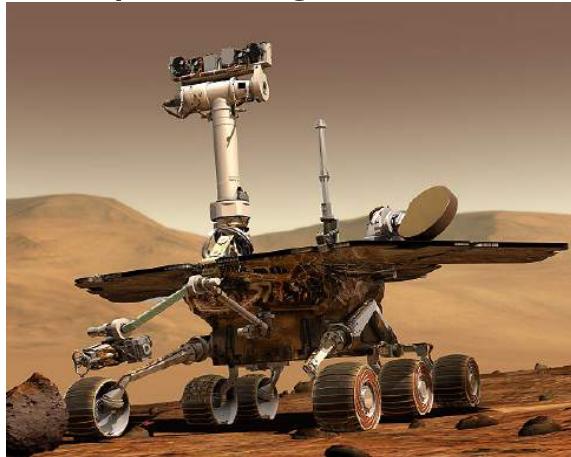
e.g. all points P, P₁, P₂, and P₃ project to the same location in the left image, but to different locations in the right image.

The right image allows us to measure how far each of these points are from the left camera (if we can solve the correspondence problem).

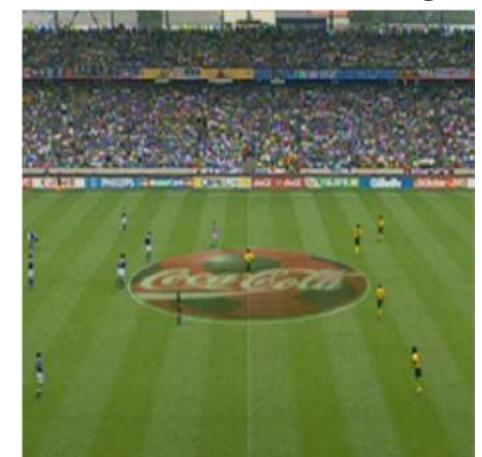
Why is stereo vision important?

This is useful for:

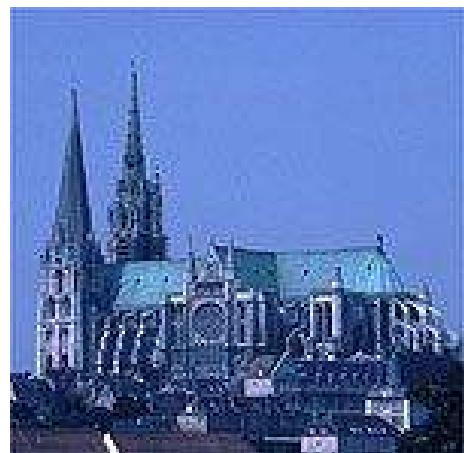
Path planning / collision avoidance (car / robot)



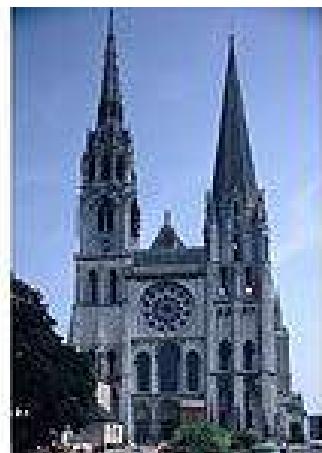
virtual advertising



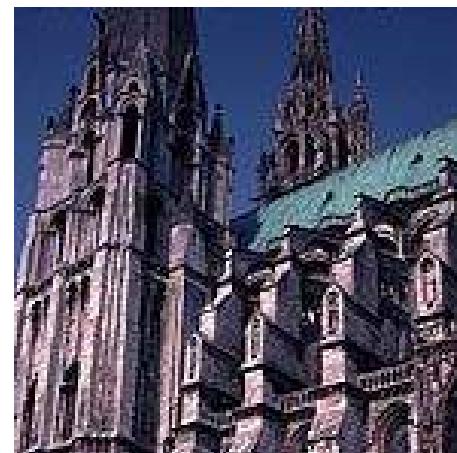
Not stereo, but same methods can be used for 3D model building



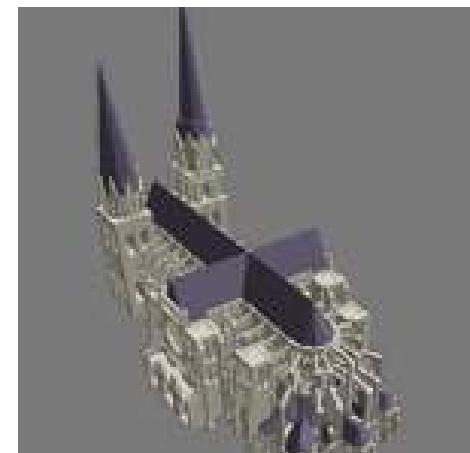
+



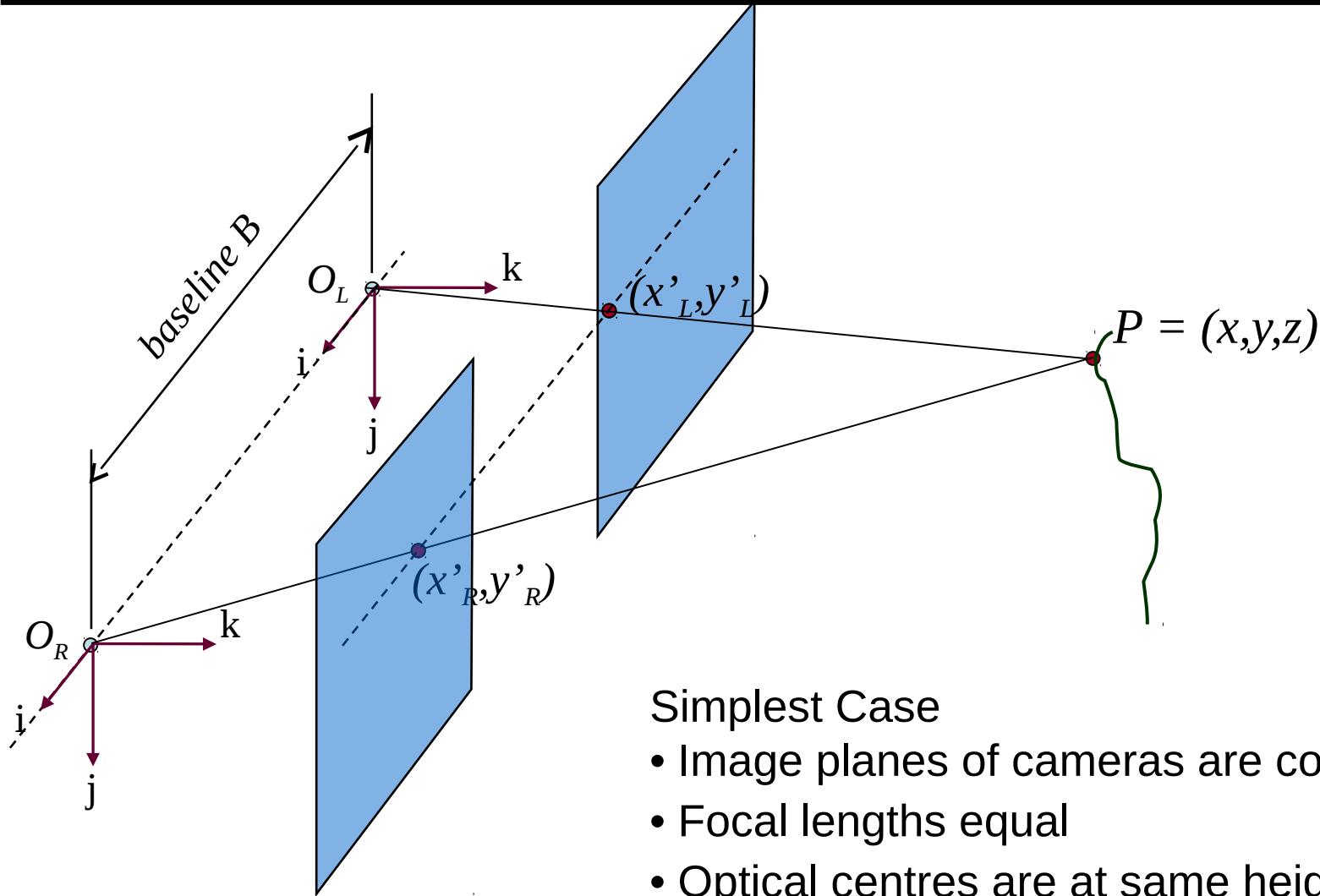
+



→



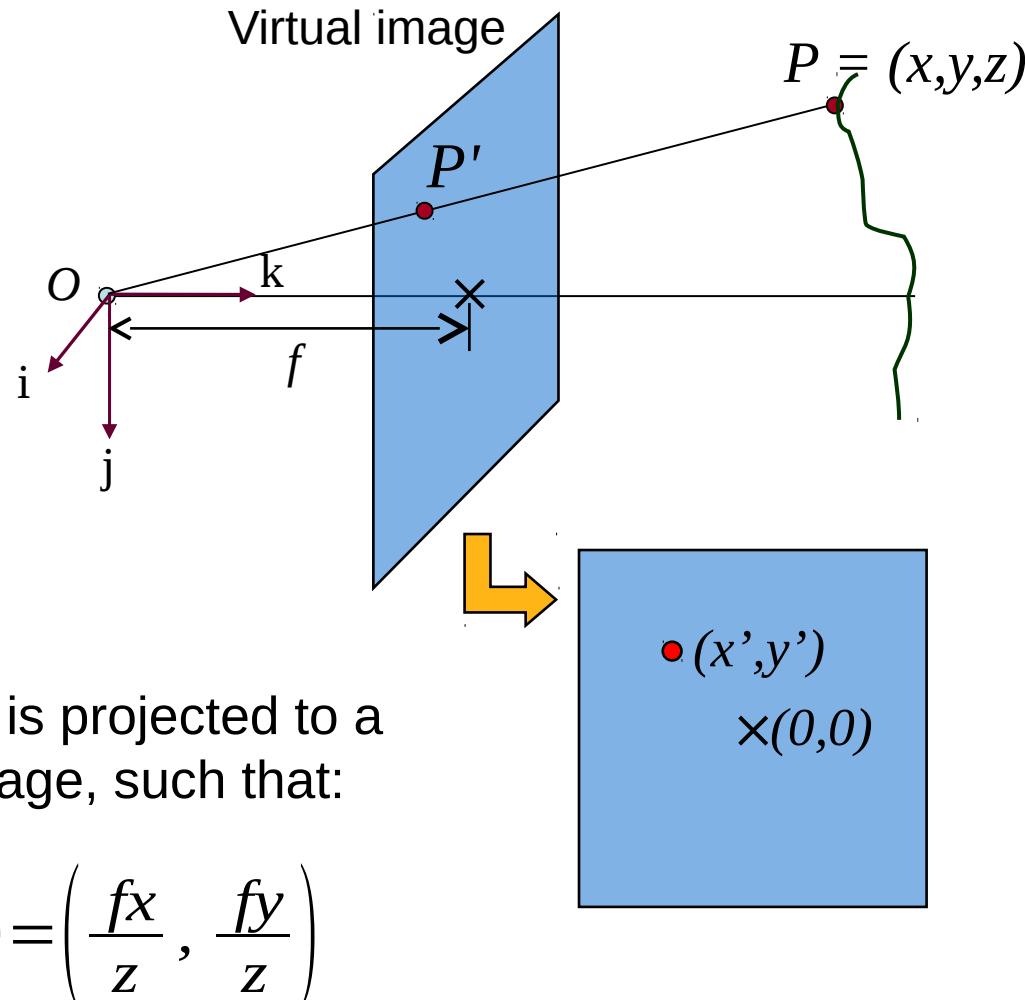
Stereo: coplanar cameras



Simplest Case

- Image planes of cameras are coplanar
- Focal lengths equal
- Optical centres are at same height (i.e. x -axes collinear)
- Intersection of optical axes at infinity (i.e. z -axes parallel)

Image formation reminder



3D scene point P is projected to a point P' on the image, such that:

$$P' = (x', y') = \left(\frac{fx}{z}, \frac{fy}{z} \right)$$

Assuming that the image centre is $(0,0)$
[see lecture 2]

Stereo: coplanar cameras

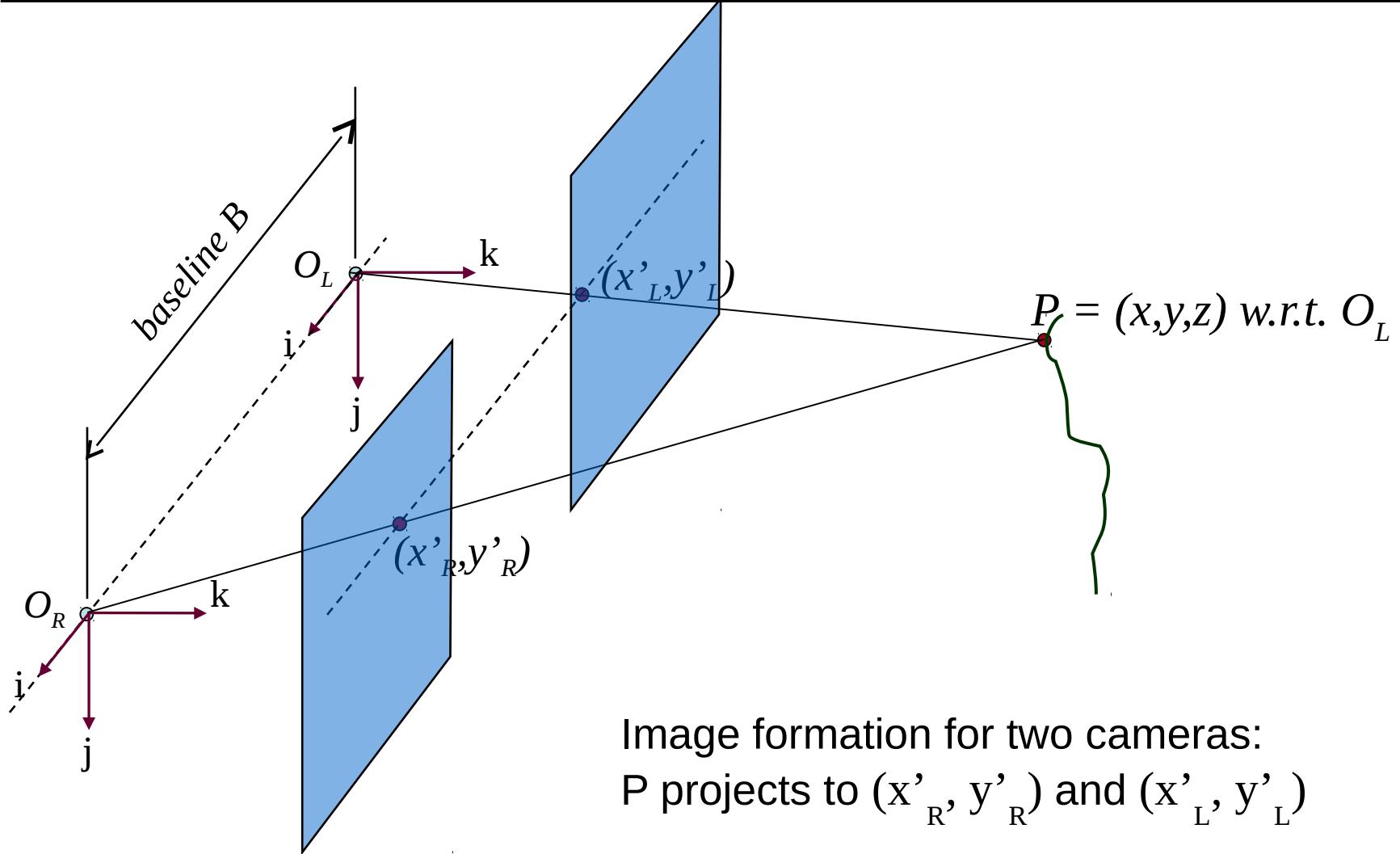
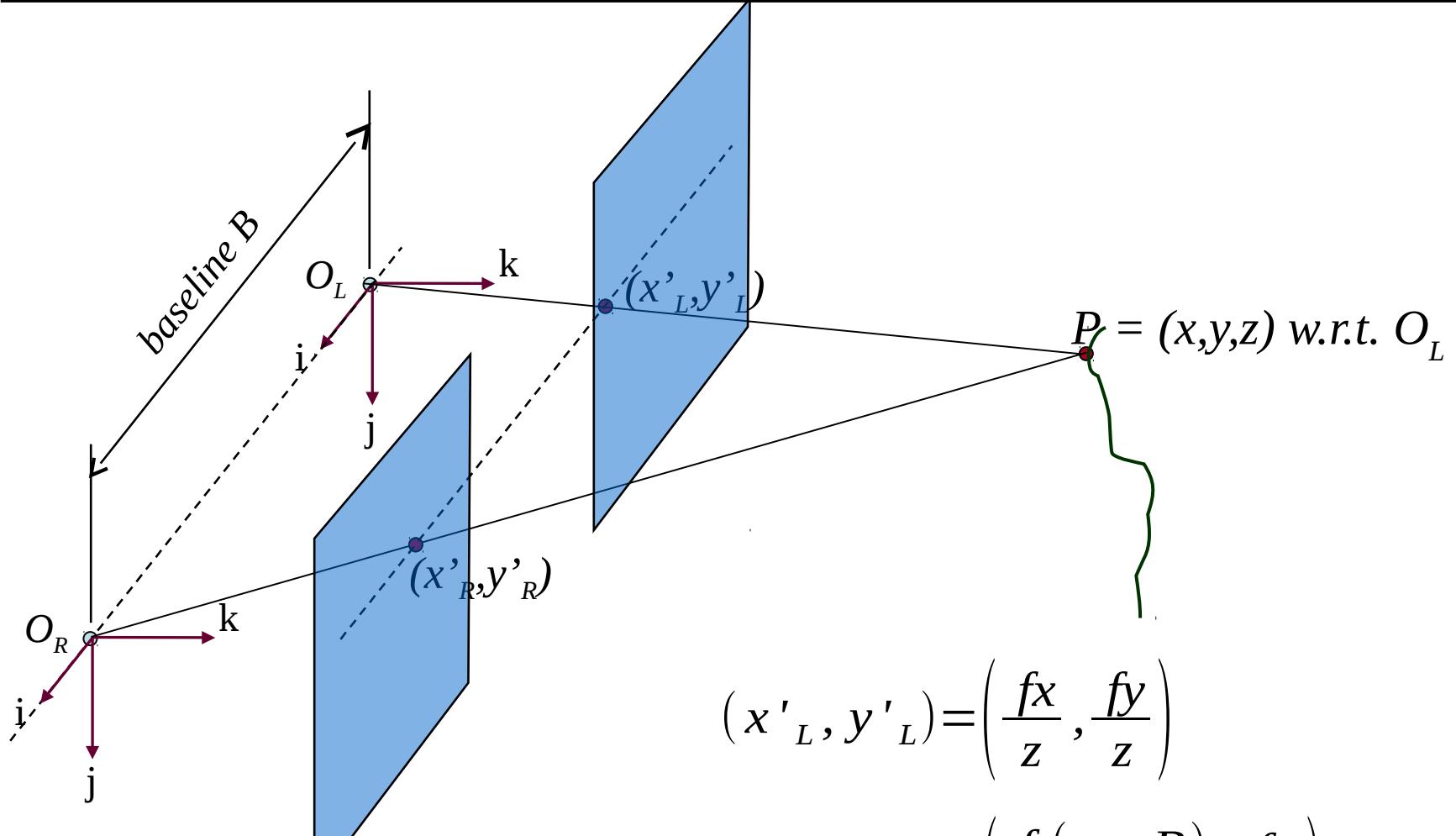


Image formation for two cameras:
 P projects to (x'_R, y'_R) and (x'_L, y'_L)

Note: Because x-axes of cameras are collinear, $y'_L = y'_R$

Stereo: coplanar cameras

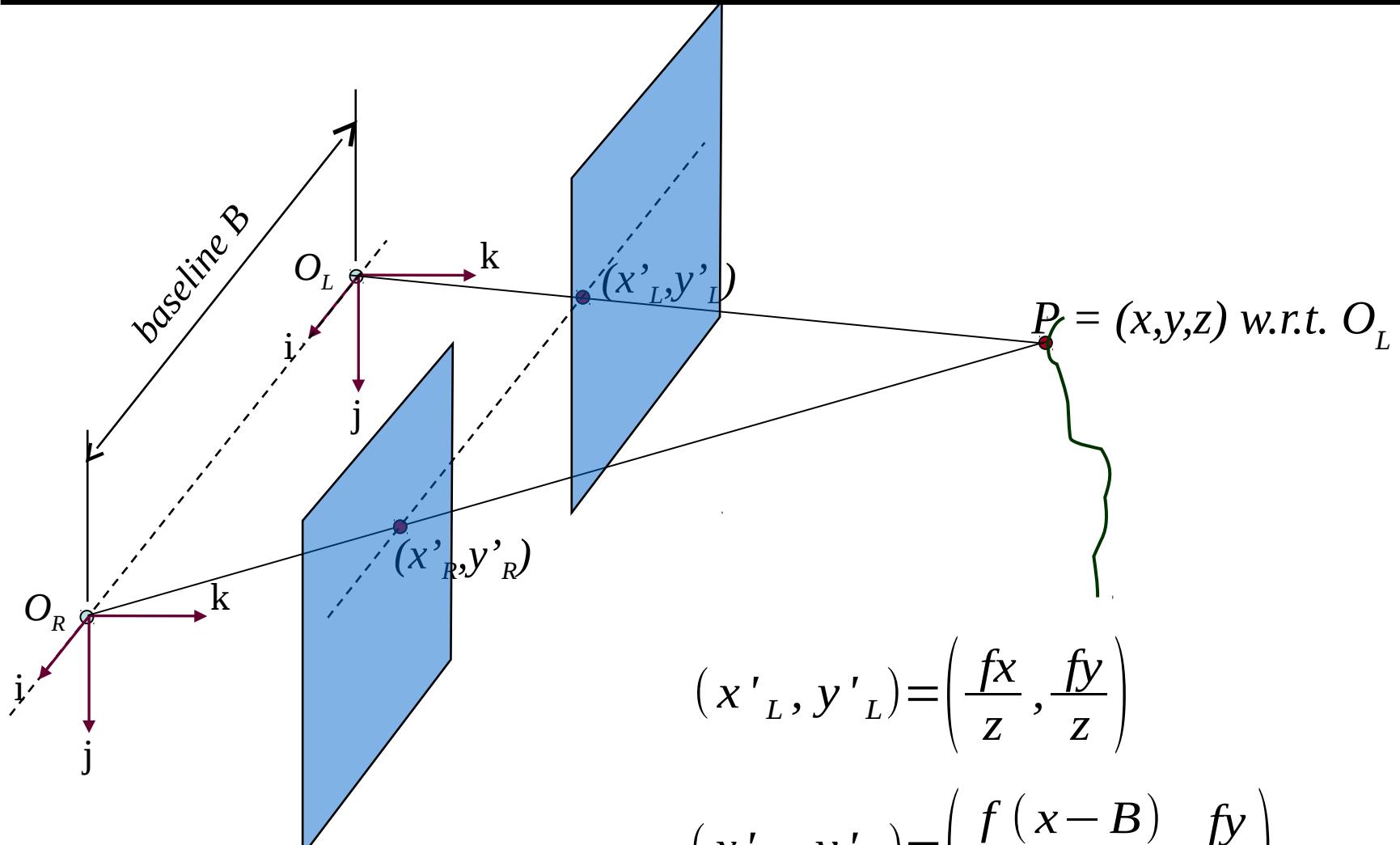


$$(x'_L, y'_L) = \left(\frac{fx}{z}, \frac{fy}{z} \right)$$

$$(x'_R, y'_R) = \left(\frac{f(x - B)}{z}, \frac{fy}{z} \right)$$

Using the coordinate system of the left camera (since $x_R = x_L - B$)

Stereo: coplanar cameras



$$\text{Disparity, } d = x'_L - x'_R = \frac{fx}{z} - \frac{f(x-B)}{z} = \frac{fB}{z} \rightarrow z = f \frac{B}{d}$$

Disparity

Depth is inversely proportional to disparity.

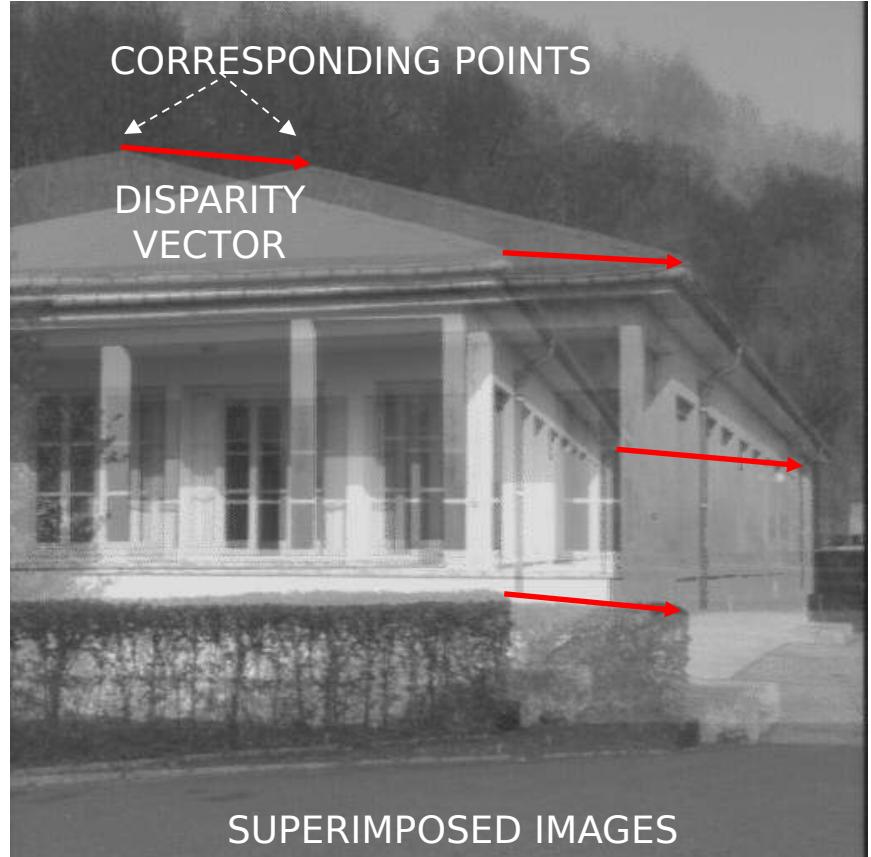
$$z = f \frac{B}{d}$$

If the baseline distance is known, and we can measure the disparity, then we can calculate the depth of a point.

Even without the baseline, we can know the relative depths of points from their relative disparities.

Disparity

The difference vector of the image coordinates of two corresponding points.



NOTE

- the disparity, d , of a point is a 2D vector.
- disparity is measured in pixels and can be positive or negative
- a pair of stereo images defines a field of disparity vectors (a disparity map)
- For coplanar cameras disparity is horizontal only

Disparity / depth map: example



Disparity / depth map: example



left image



right image



False shallow region
caused by false
matches

depth map
light = close, dark = far

The stereo correspondence problem

To measure disparity, it is necessary to find corresponding points in the stereo pair of images.

To solve the stereo correspondence problem, we can use:

- **Correlation-based methods**
yield dense disparity maps: a disparity value at each pixel.
- **Feature-based methods**
yield sparse disparity maps: a disparity value at interest points only.

The stereo correspondence problem

To measure disparity, it is necessary to find corresponding points in the stereo pair of images.

Basic requirements to be able to solve the correspondence problem:

1. Most scene points visible in both images
2. Corresponding image regions appear “similar”

These assumptions hold if:

- The distance of the 3D point from the cameras is much larger than the baseline: $z \gg B$

The stereo correspondence problem

To measure disparity, it is necessary to find corresponding points in the stereo pair of images.

As we saw in the previous lecture, solving the correspondence problem is not easy.

However, we can use knowledge about the stereo camera system to help find a solution...

Stereo Constraints on Correspondence

Epipolar constraint

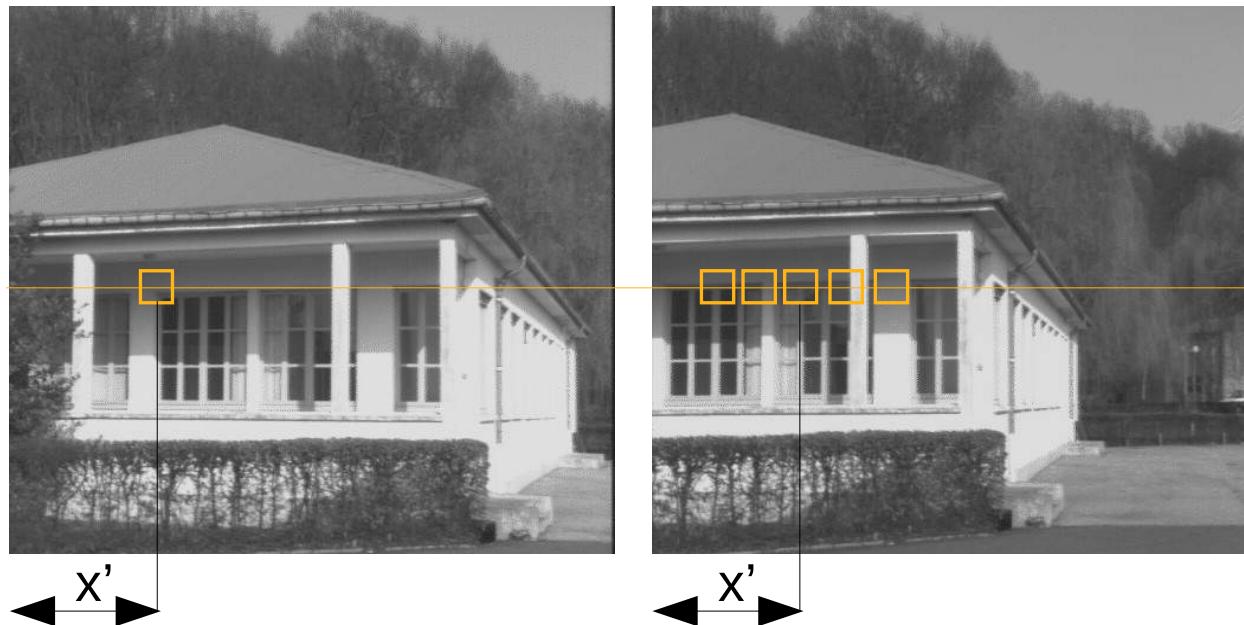
For coplanar cameras, $y'_L = y'_R$ so 2D search can be reduced to a 1D search along the “epipolar” line (= the corresponding row of pixels for coplanar cameras).



Stereo Constraints on Correspondence

Maximum disparity constraint

Length of search region depends on the maximum expected disparity, often predictable geometrically ($d_{max} = fB/z_{min}$).

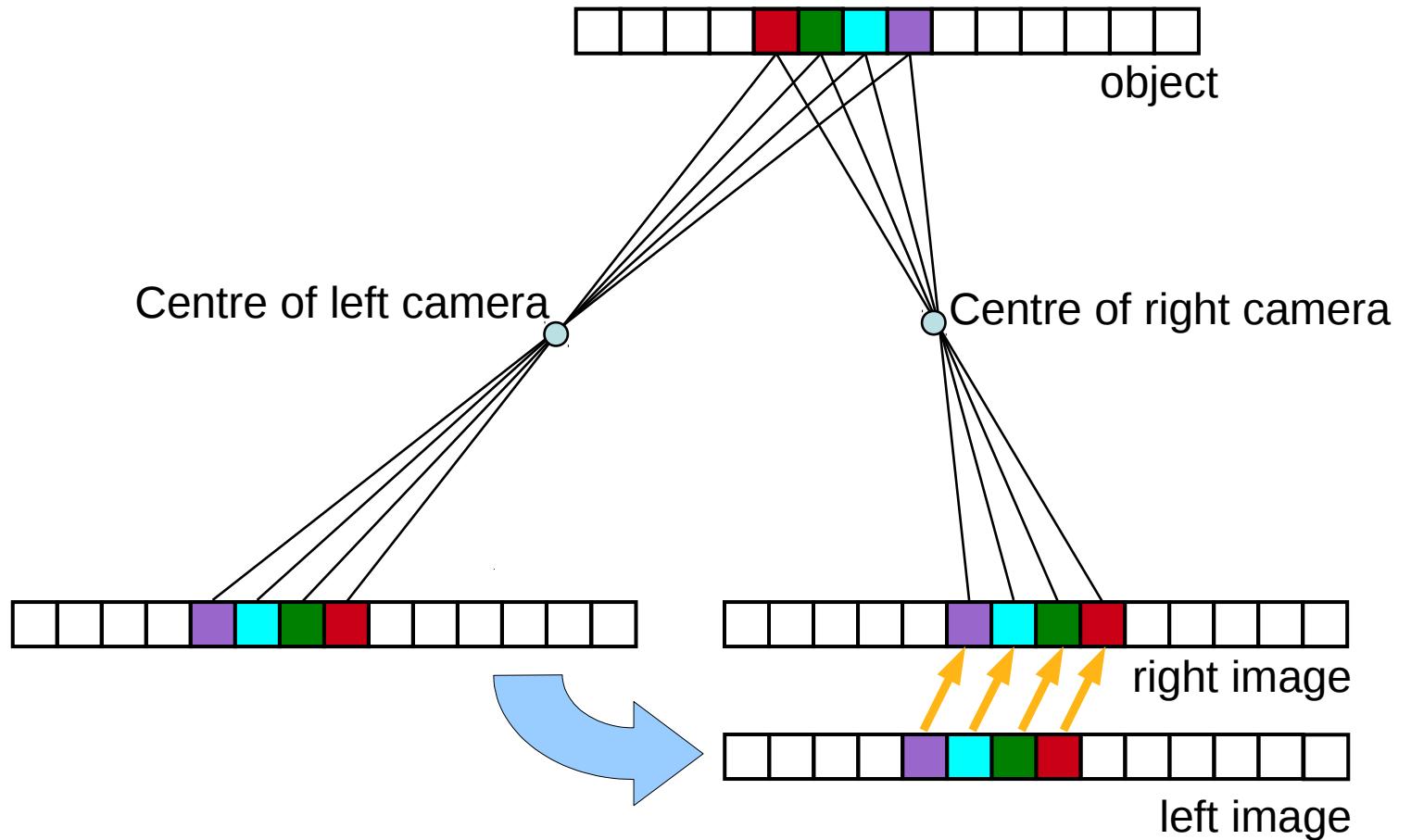


For each point (x', y') in the left image, search for its corresponding point between $(x' - d_{max}, y')$ and $(x' + d_{max}, y')$ in the right image.

Stereo Constraints on Correspondence

Continuity

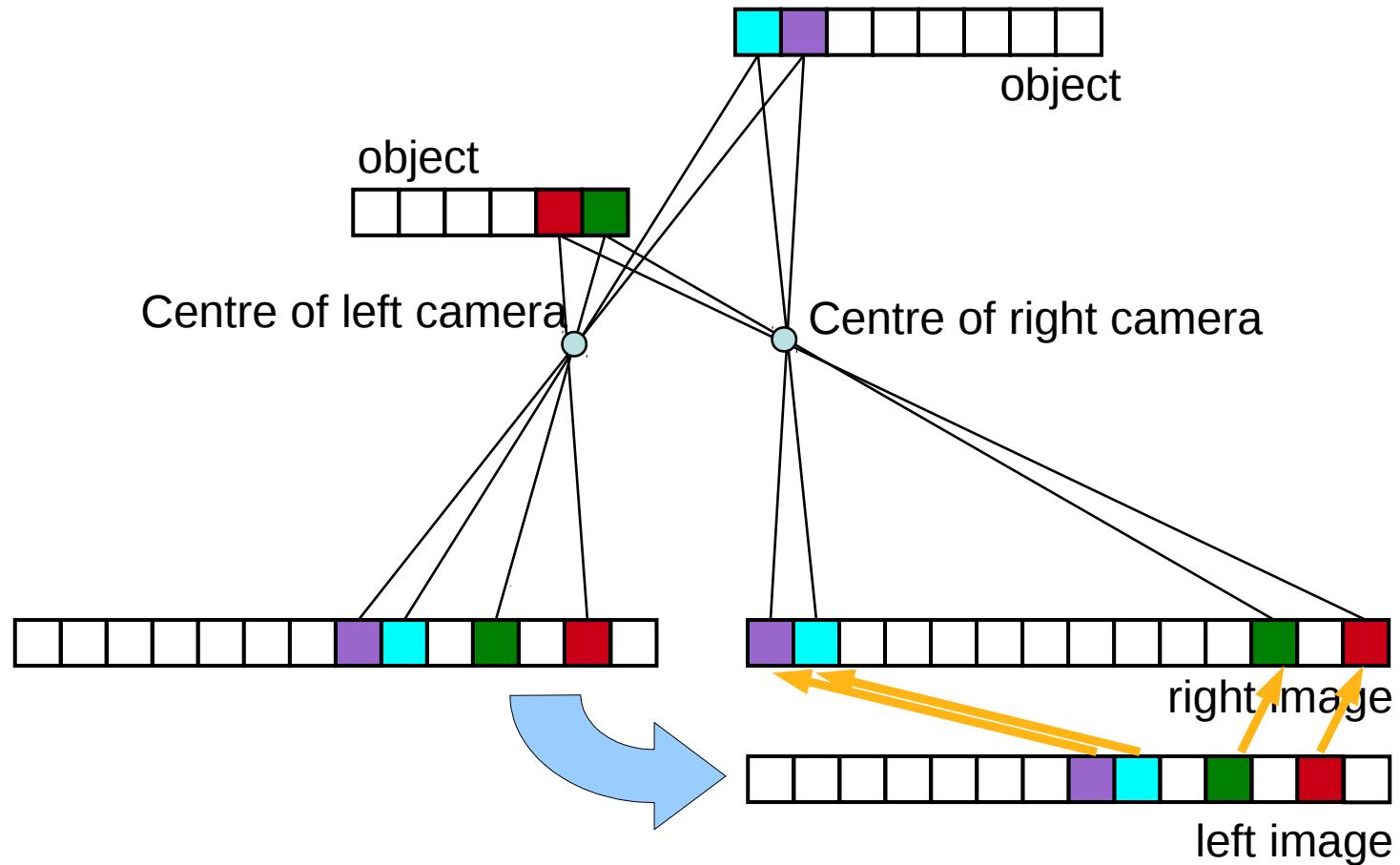
Neighbouring points should have similar disparities, because the environment is made of continuous surfaces over which depth varies smoothly.



Stereo Constraints on Correspondence

Continuity

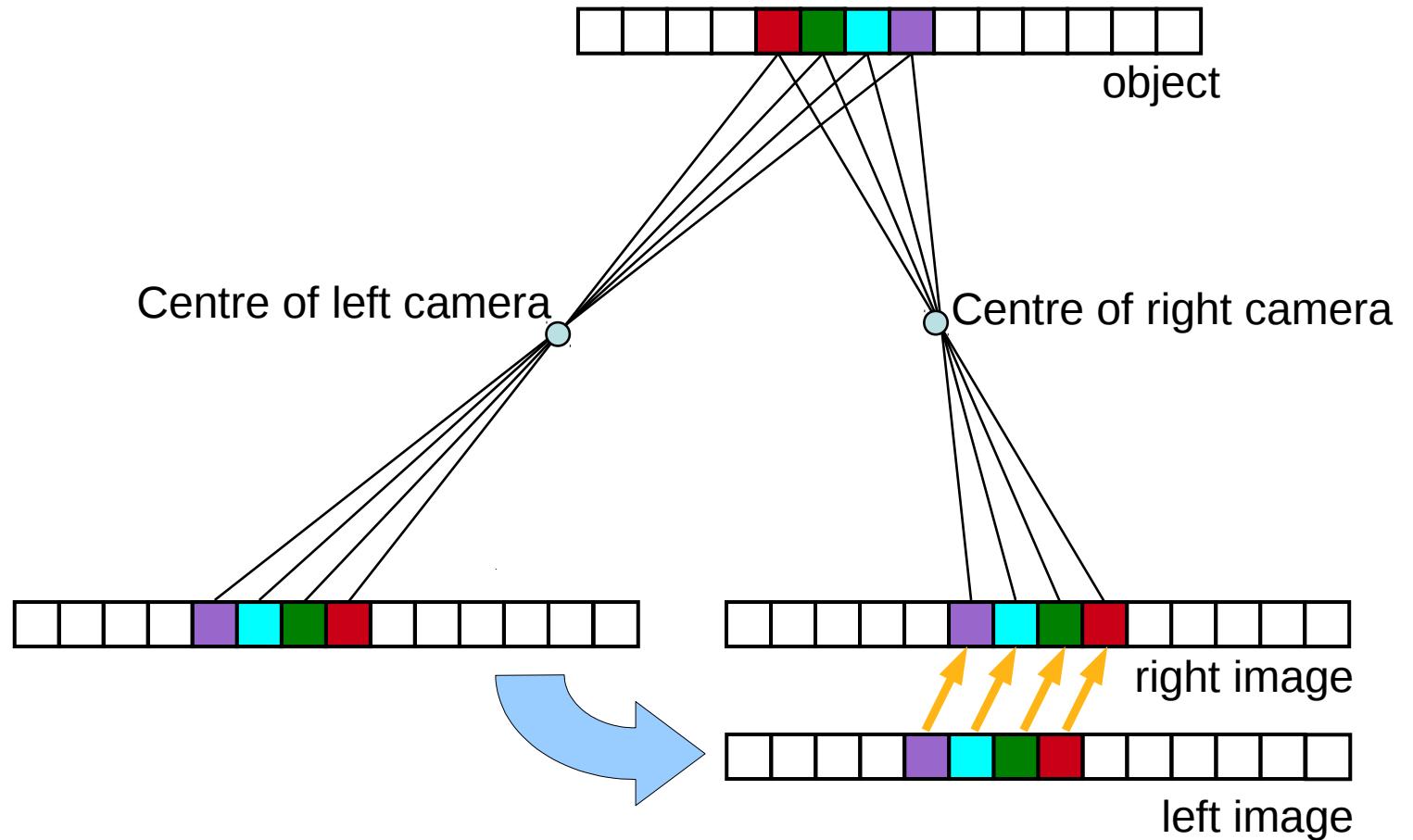
The exception is at discontinuities where depth (and hence disparity) can change suddenly.



Stereo Constraints on Correspondence

Uniqueness

A location in one image should only match a single location in the other image.

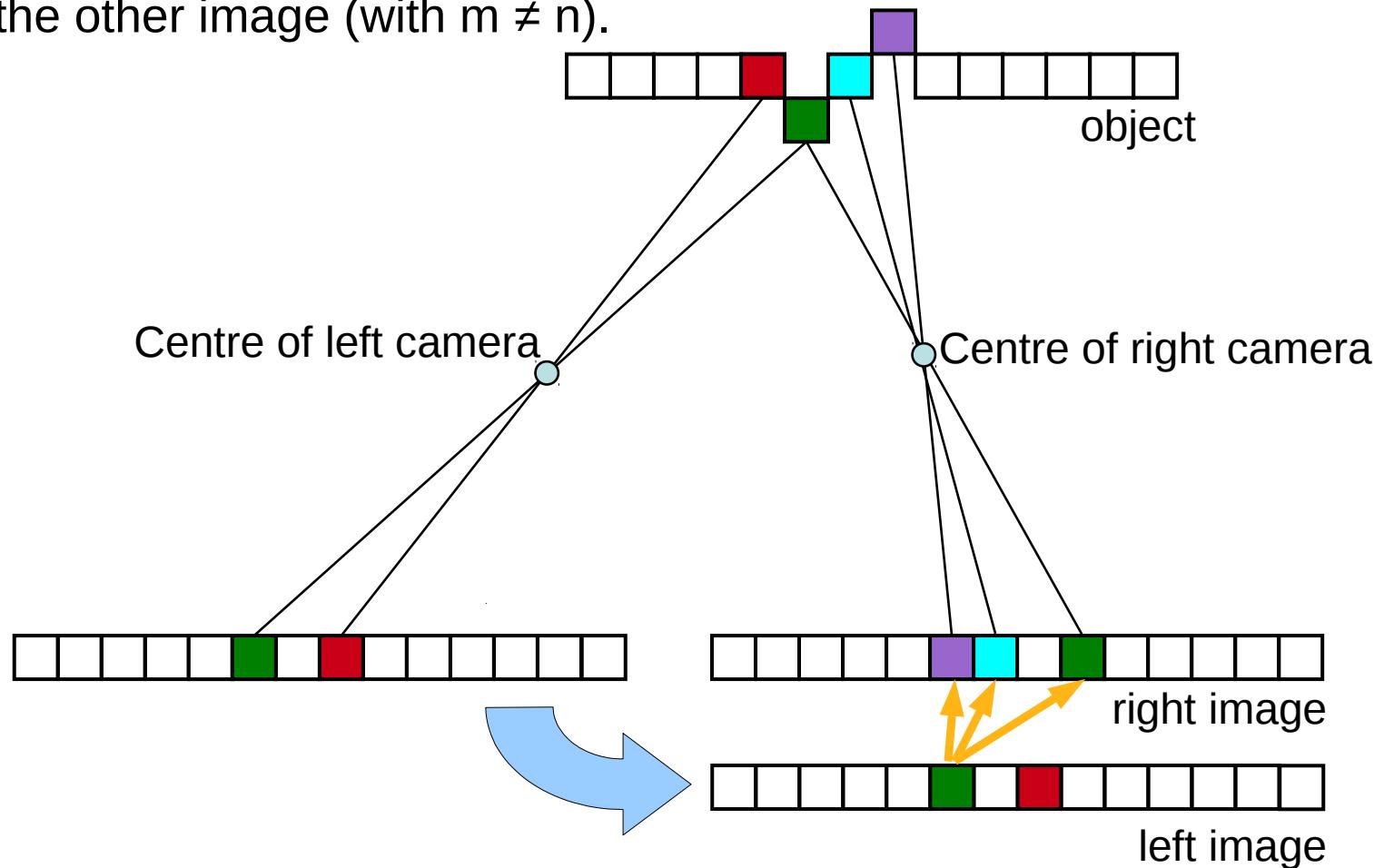


Stereo Constraints on Correspondence

Uniqueness

The exception is when a surface lies along a line-of-sight for one camera (in this case one location may match many locations).

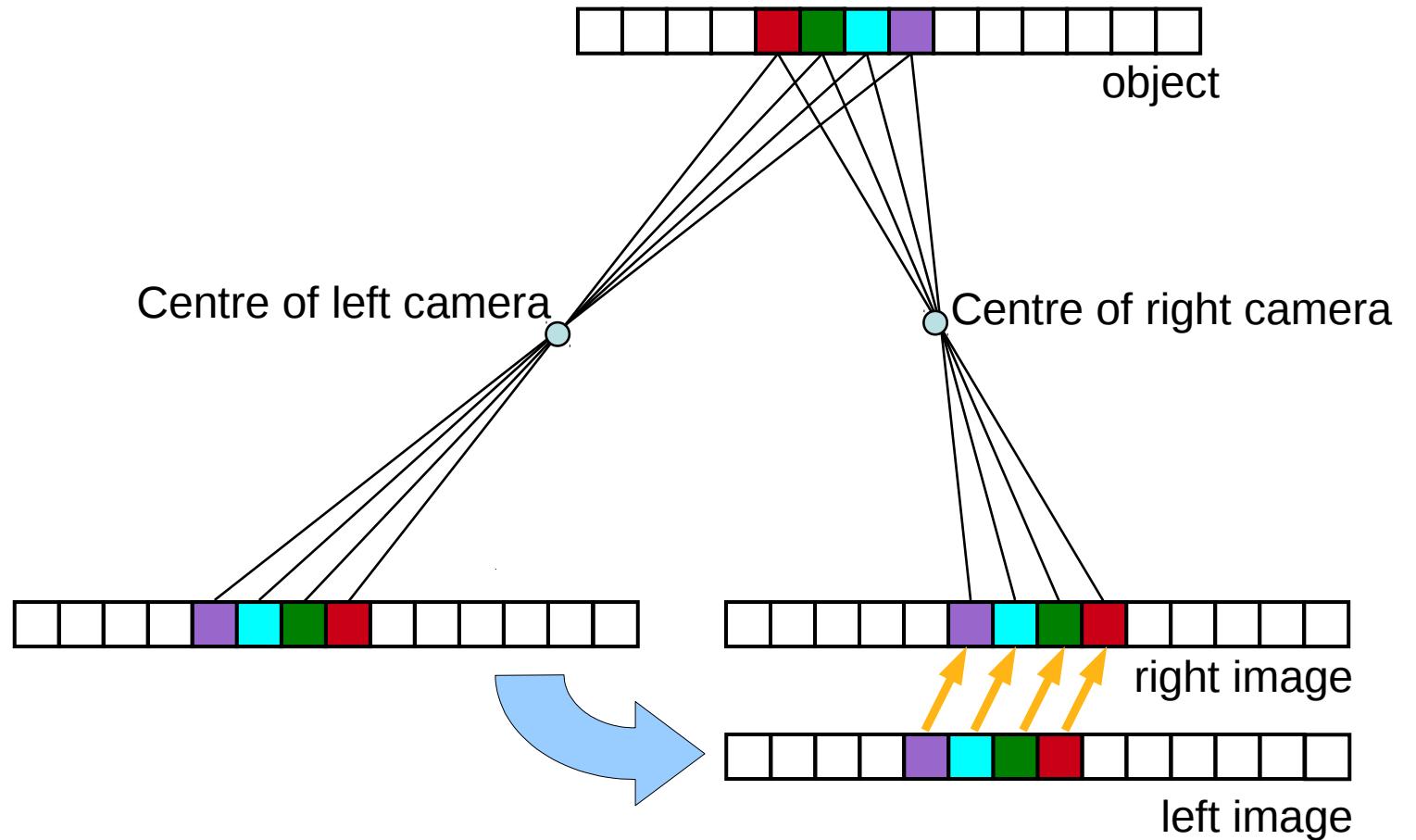
In fact any inclined surface may project to n pixels in one image and m pixels in the other image (with $m \neq n$).



Stereo Constraints on Correspondence

Ordering

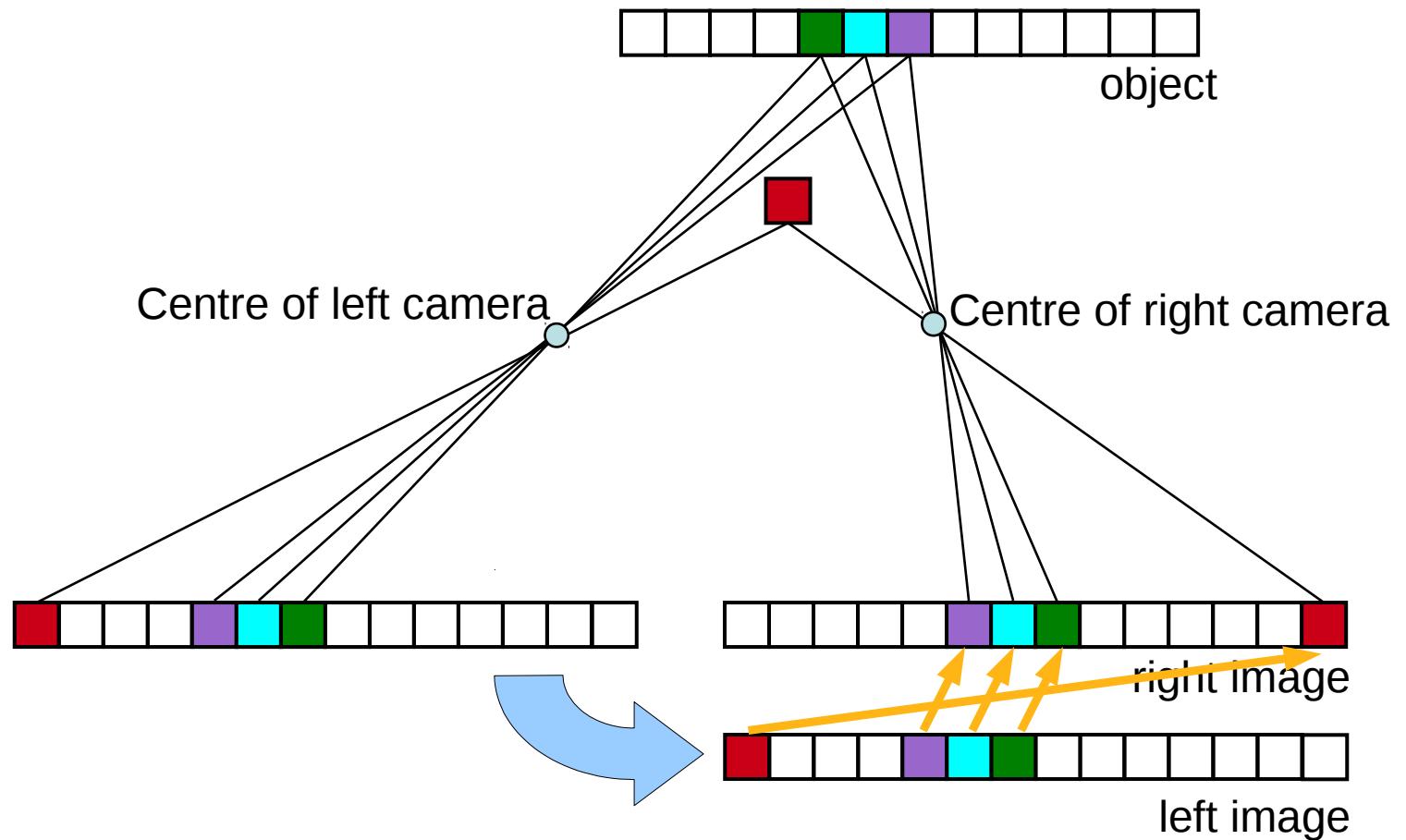
Matching points along corresponding epipolar lines should be in the same order.



Stereo Constraints on Correspondence

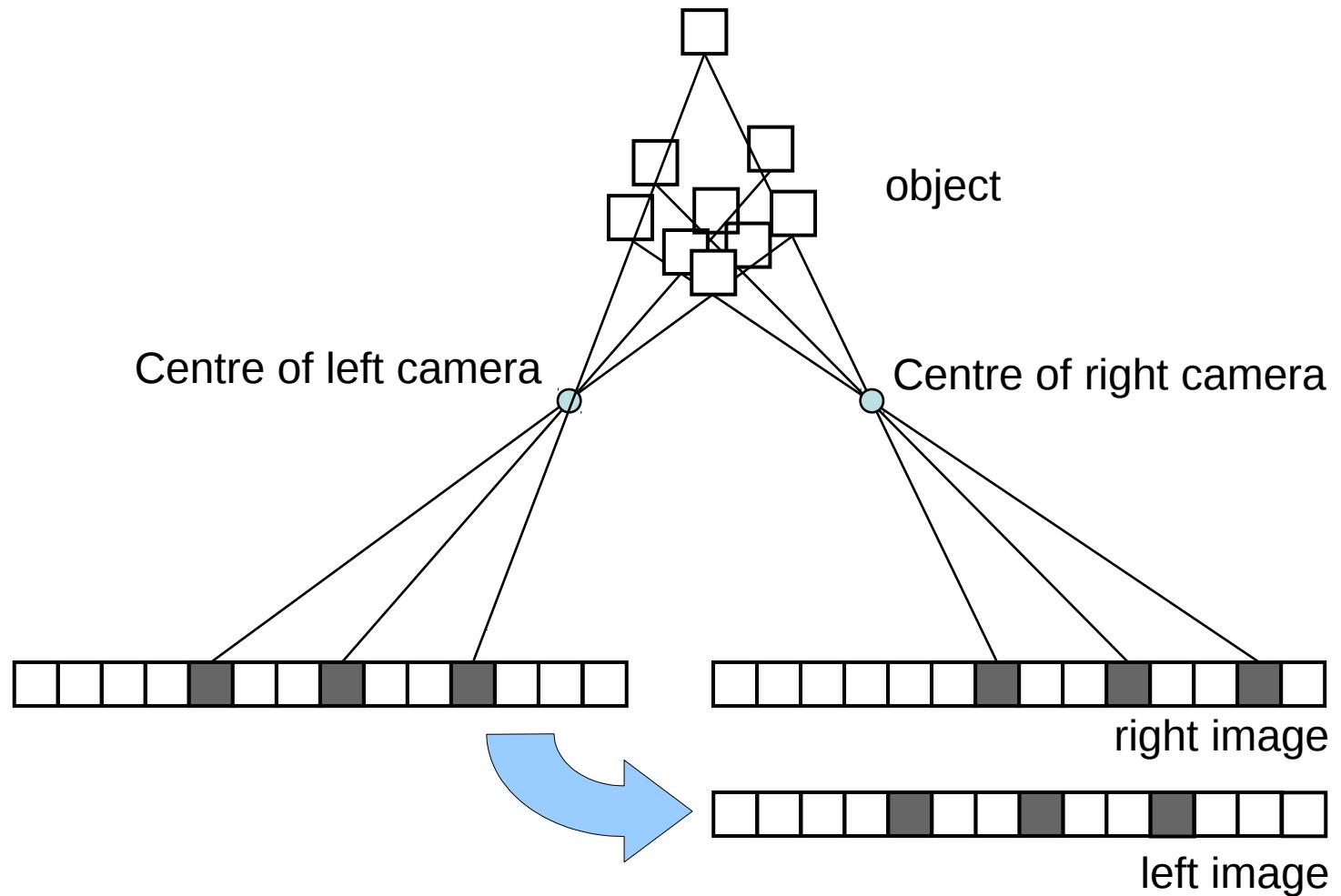
Ordering

The exception is when objects have different depths.



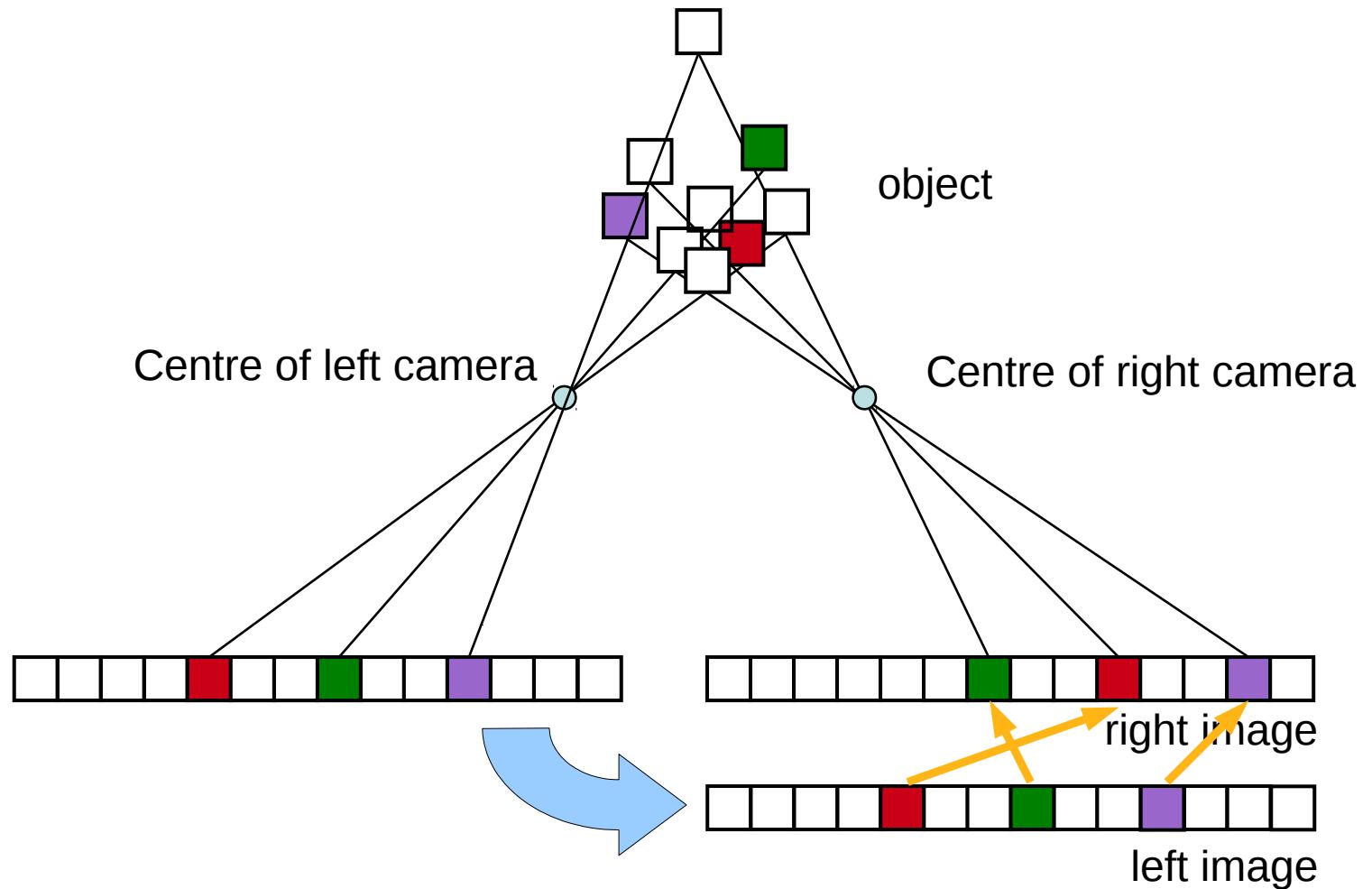
Correspondence problems

Correspondence is fundamentally ambiguous, i.e. there are many possible solutions.



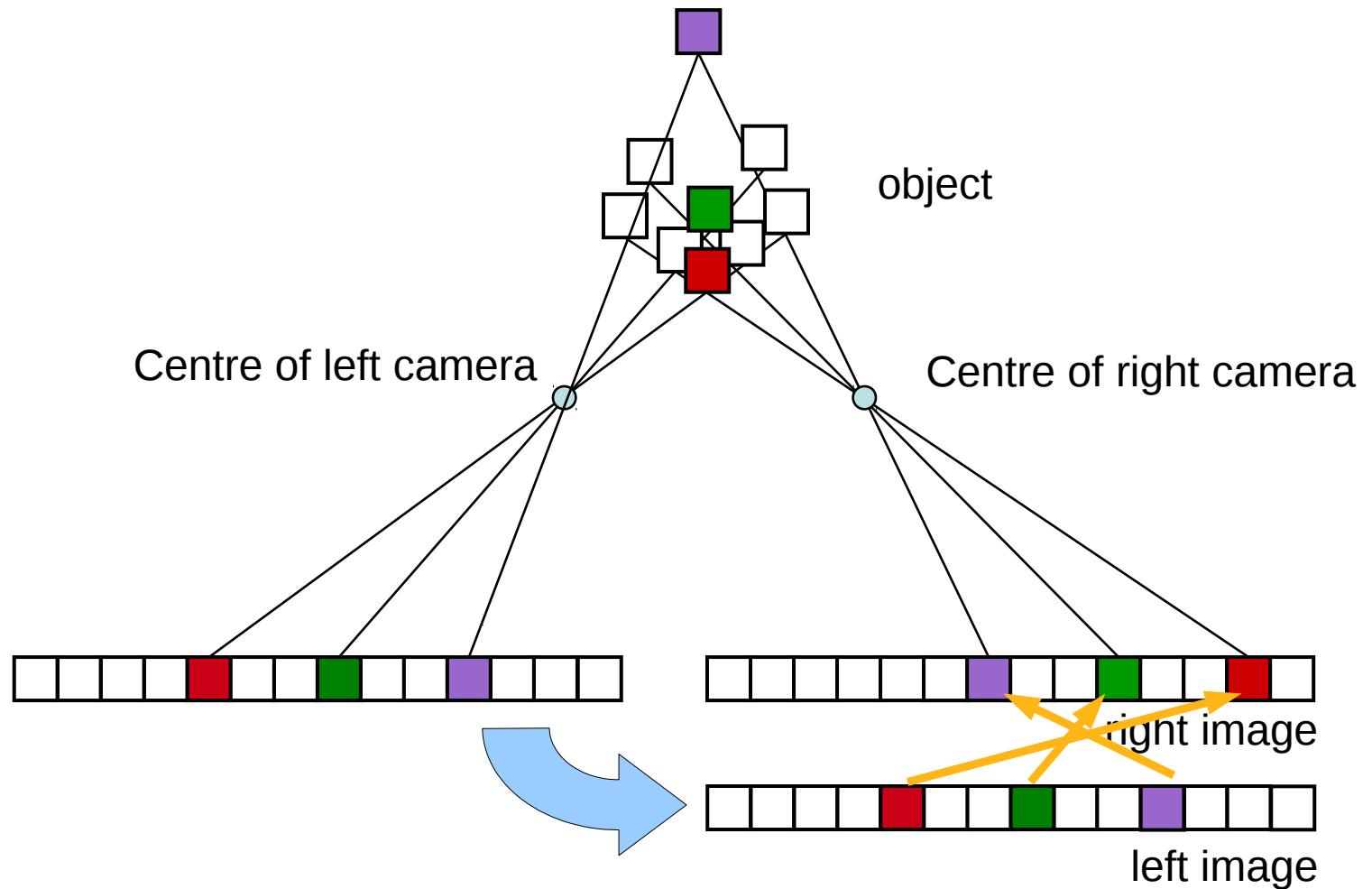
Correspondence problems

Correspondence is fundamentally ambiguous, i.e. there are many possible solutions.



Correspondence problems

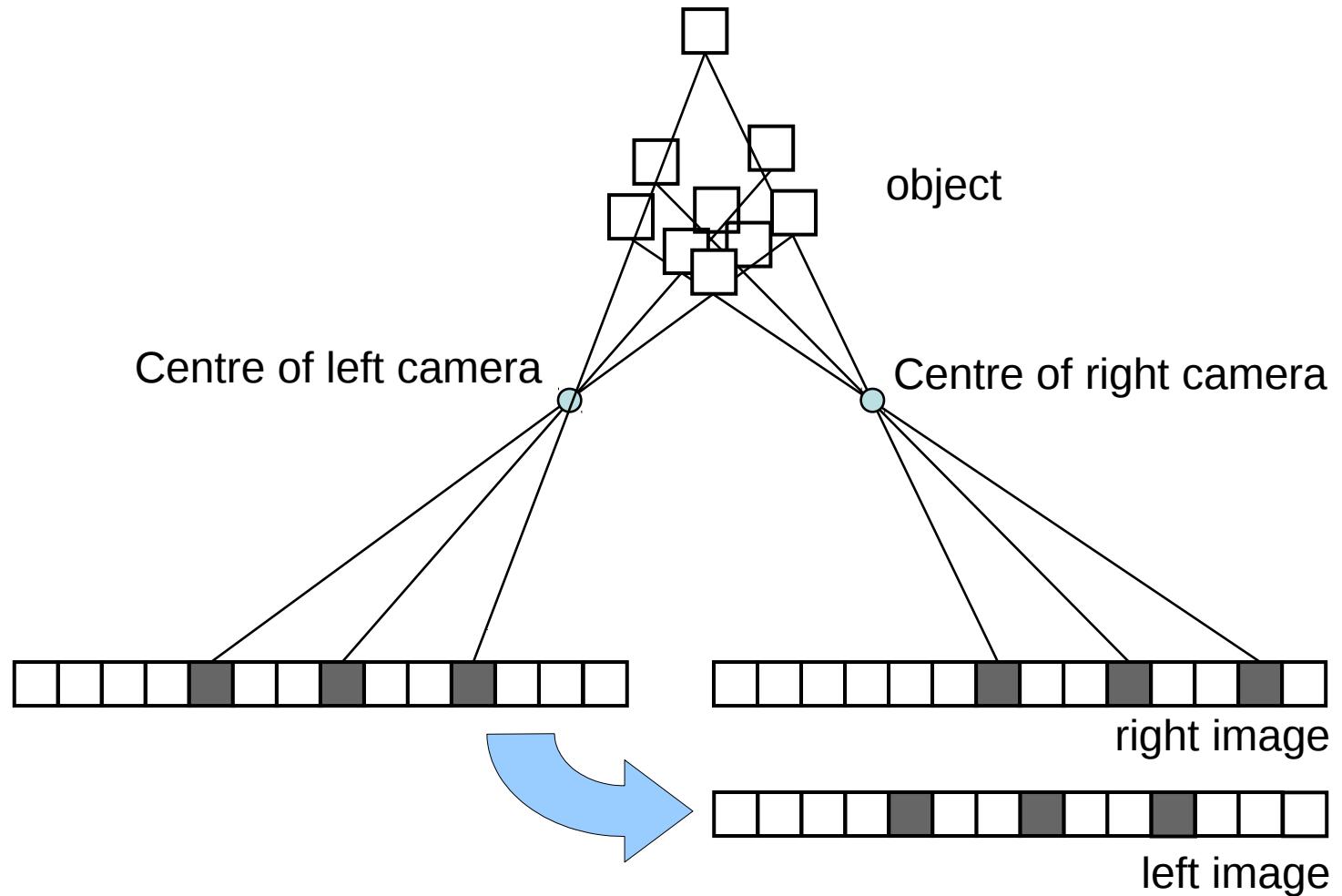
Correspondence is fundamentally ambiguous, i.e. there are many possible solutions.



Correspondence problems

Correspondence is fundamentally ambiguous, i.e. there are many possible solutions.

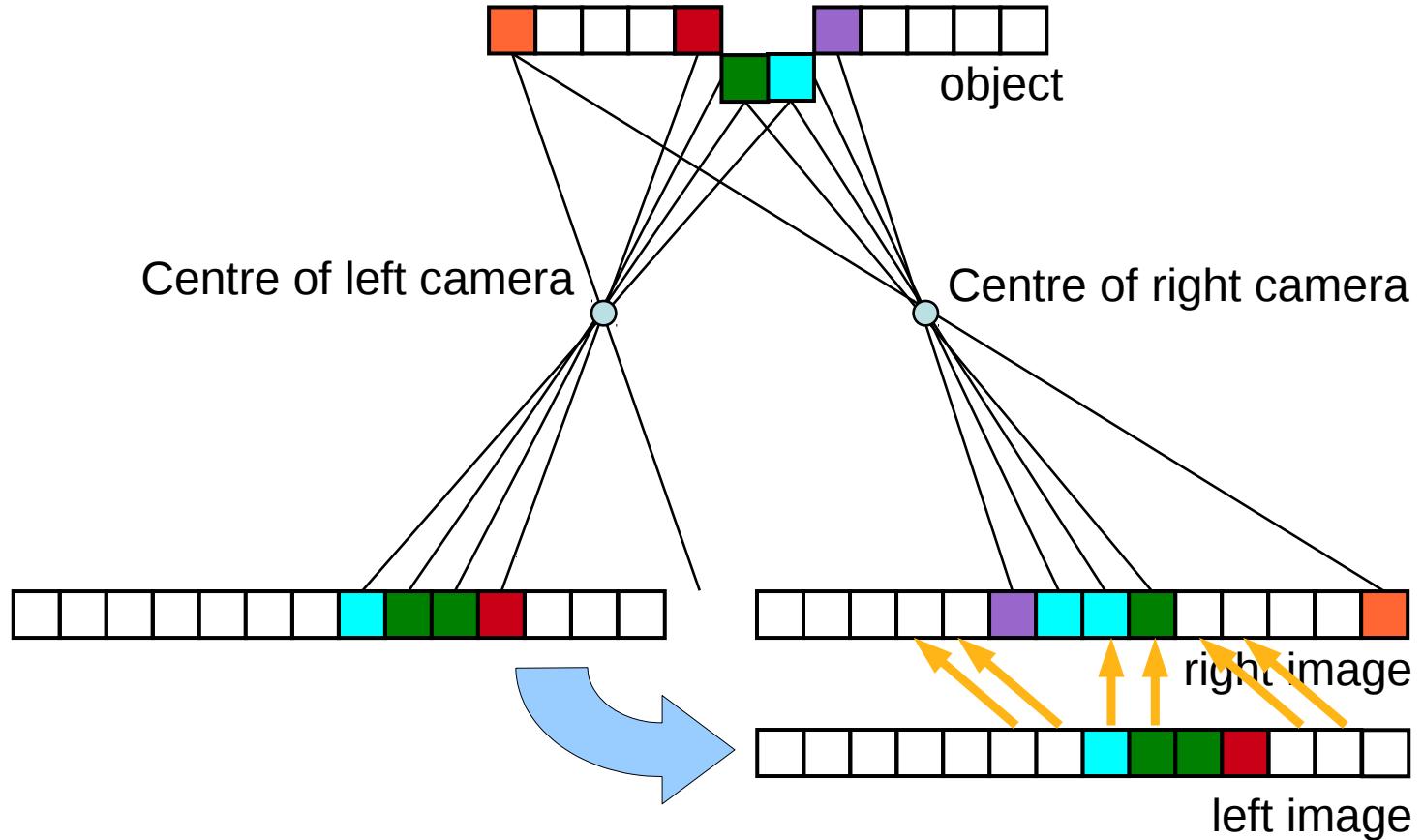
We are trying to use imperfect constraints to narrow down these many potential solutions to the correct one.



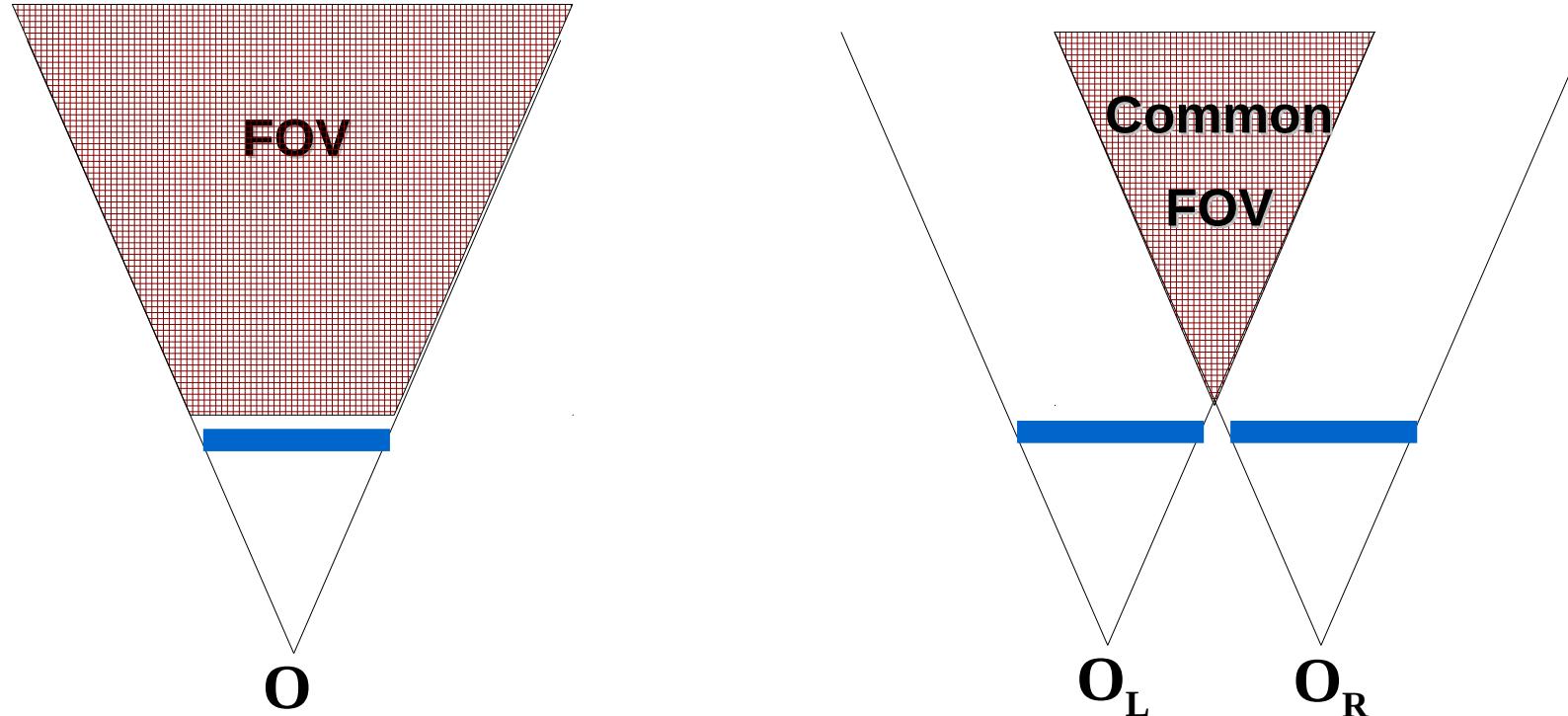
Correspondence problems

Some points in each image will have no corresponding points in the other image:

1. due to occlusion (e.g.  )
2. the cameras might have different fields of view (e.g. )



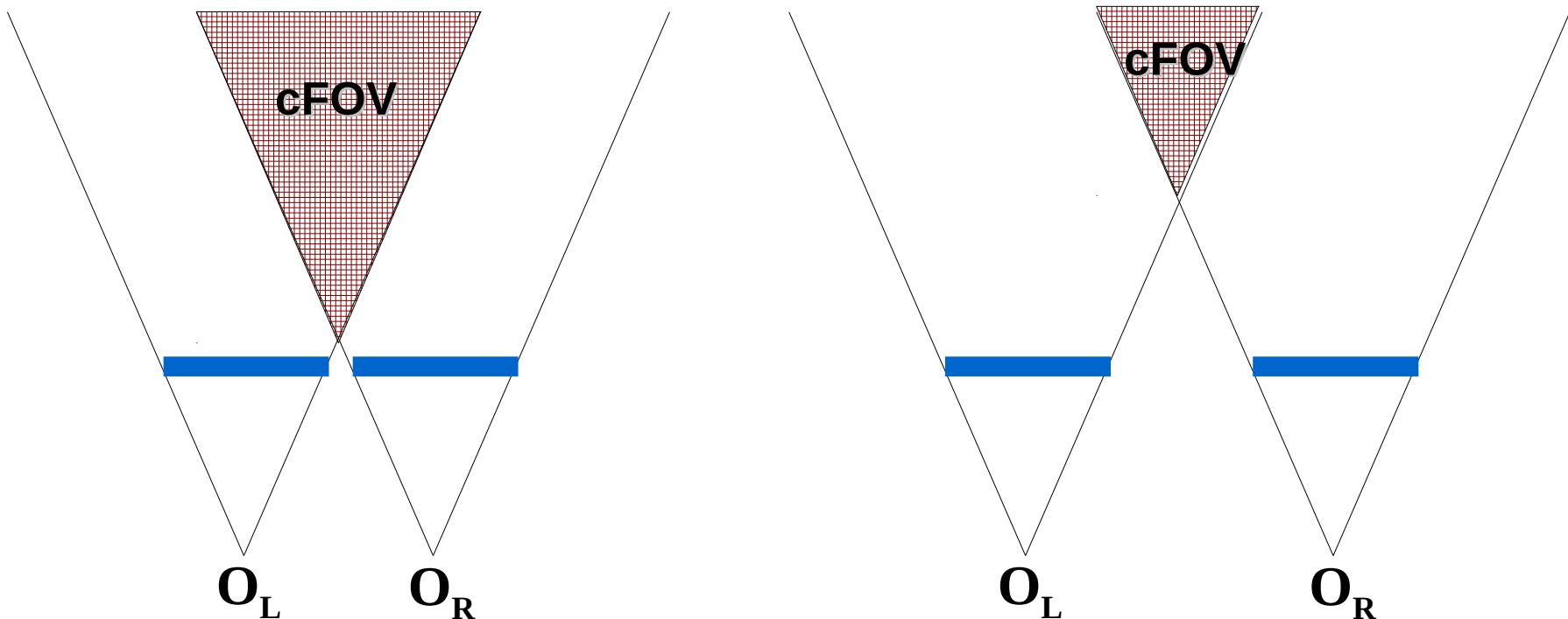
Field of view



One camera can
only take an image
of those locations
within its field of view
(FOV)

A stereo pair of
cameras can find
depth for those
locations within the
common FOV of the
two cameras

Stereo: coplanar cameras



Short baseline

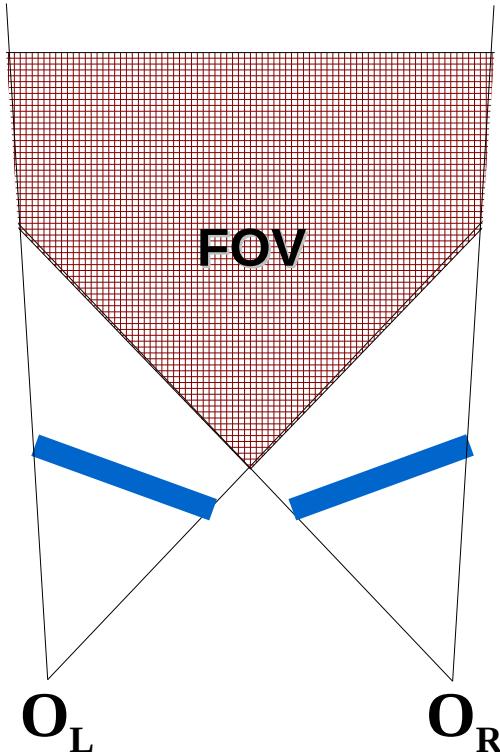
- large common FOV
- large depth error
(changes in depth cause only small changes in disparity)

$$z = f \frac{B}{d}$$

Long baseline

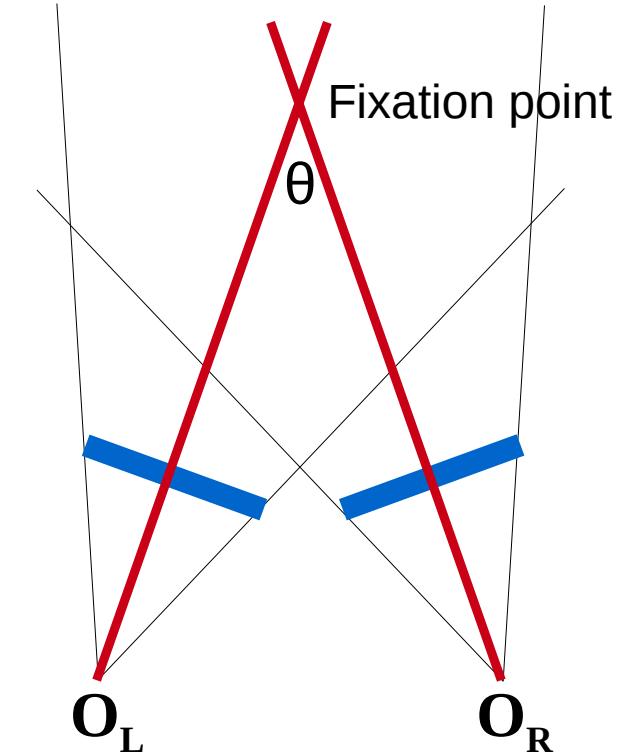
- small common FOV
- small depth error
(changes in depth cause larger changes in disparity)

Stereo: non-coplanar cameras



Intersecting optical axes

- large common FOV
- small depth error



Convergence angle θ
("vergence")

Stereo: non-coplanar cameras

Disparity measured using angles instead of distance

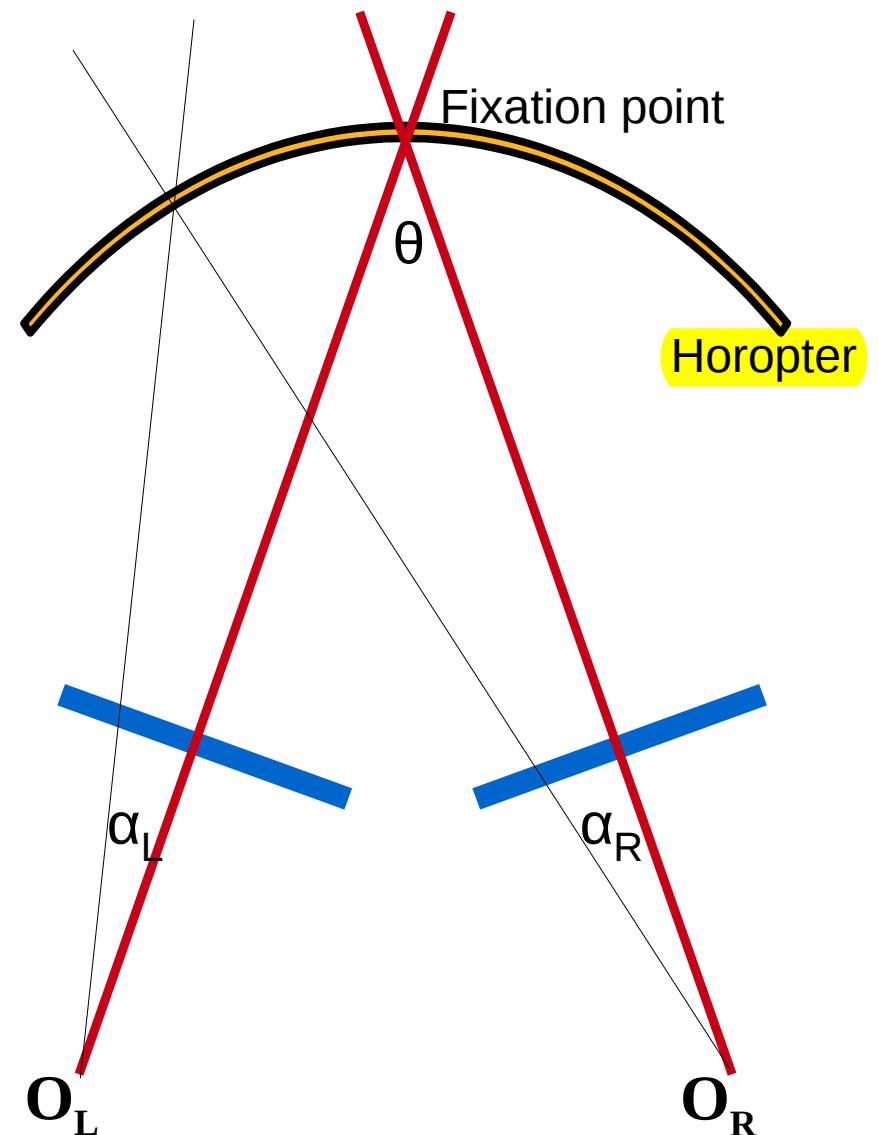
$$\text{Disparity} = \alpha_L - \alpha_R$$

Disparity = zero at fixation point

Disparity = zero at all points on a curve where rays at equal angles intersect.

Curve with zero-disparity called the “horopter”

Location of horopter depends on vergence angle.



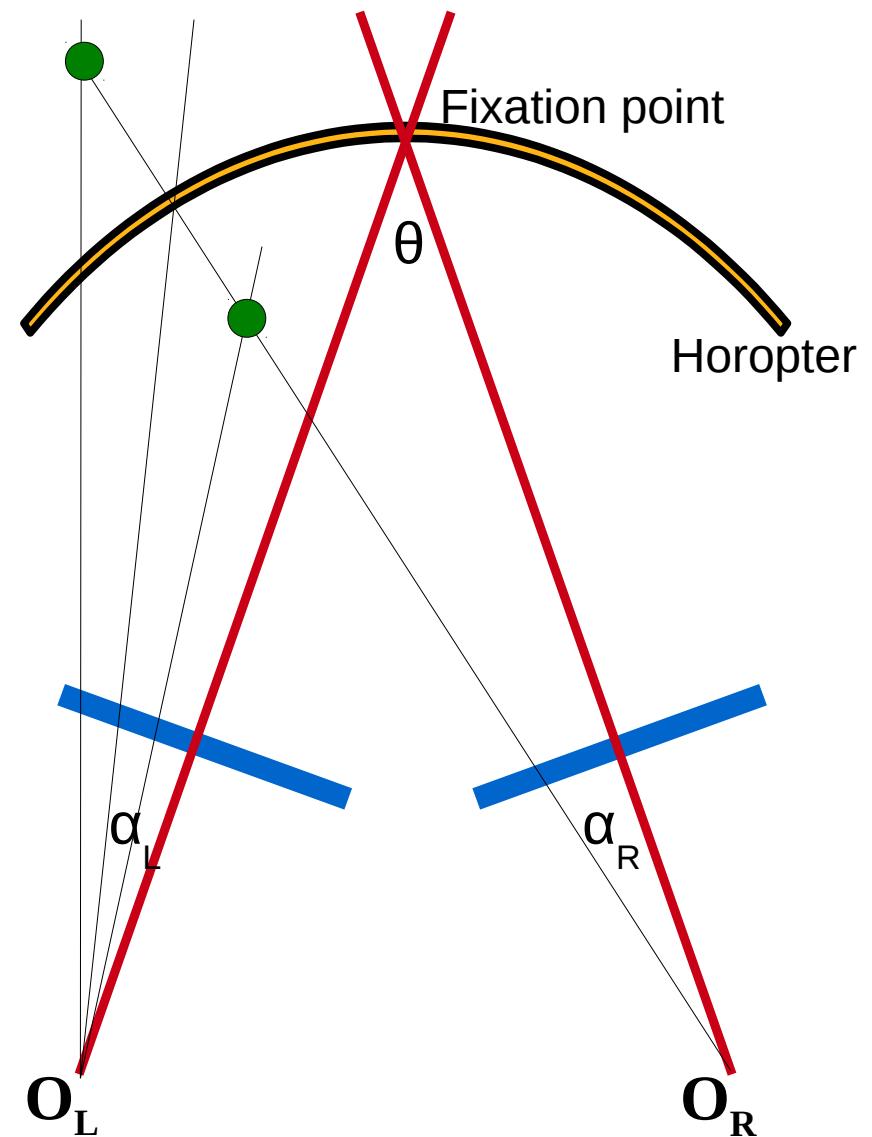
Stereo: non-coplanar cameras

Magnitude of disparity increases with the distance of objects from the horopter

$(\alpha_L - \alpha_R) > 0$: outside of the horopter

$(\alpha_L - \alpha_R) < 0$: inside the horopter

Need to be consistent with signs of angles, and order of subtraction.

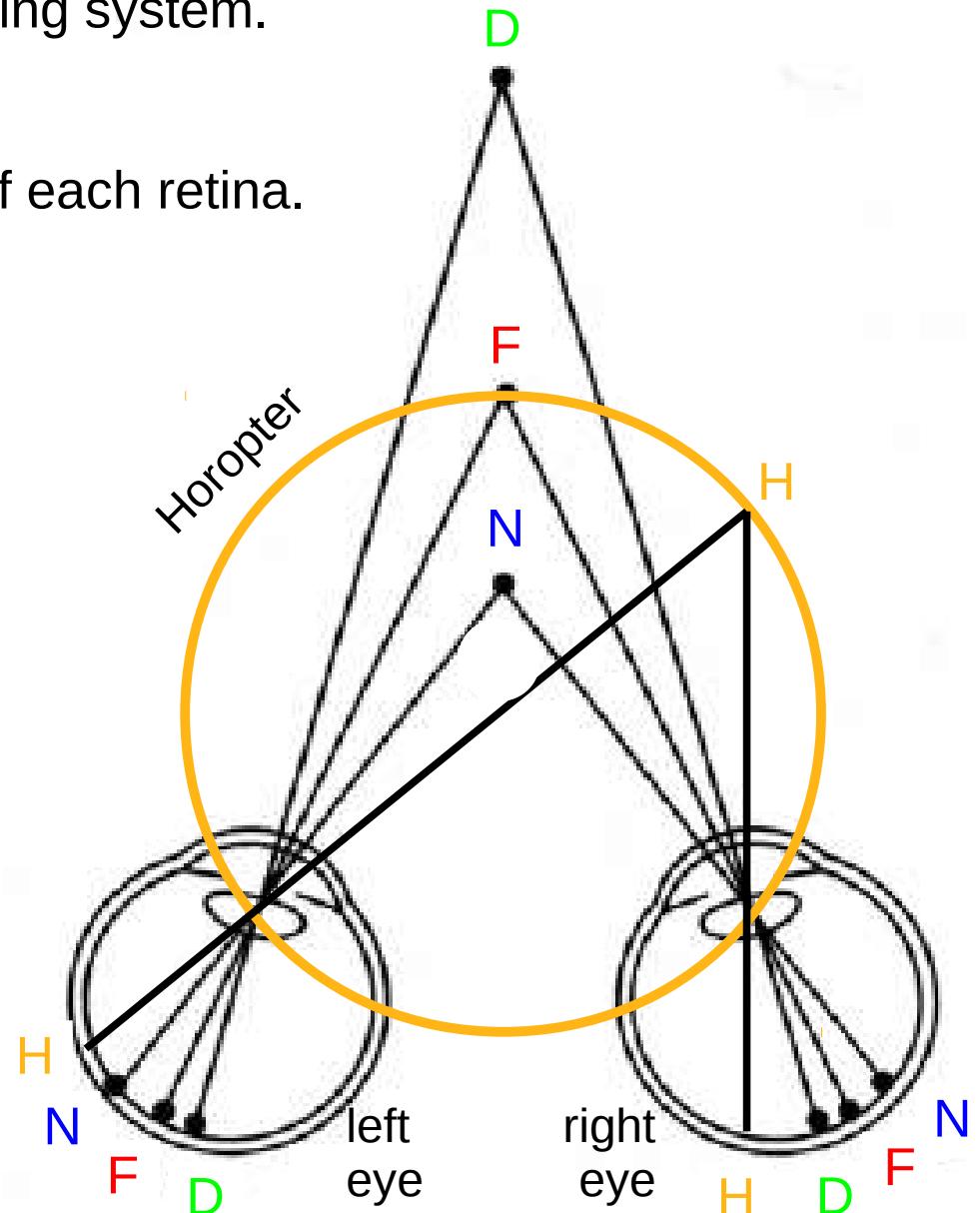


Stereo: non-coplanar eyes

Humans employ a non-coplanar imaging system.

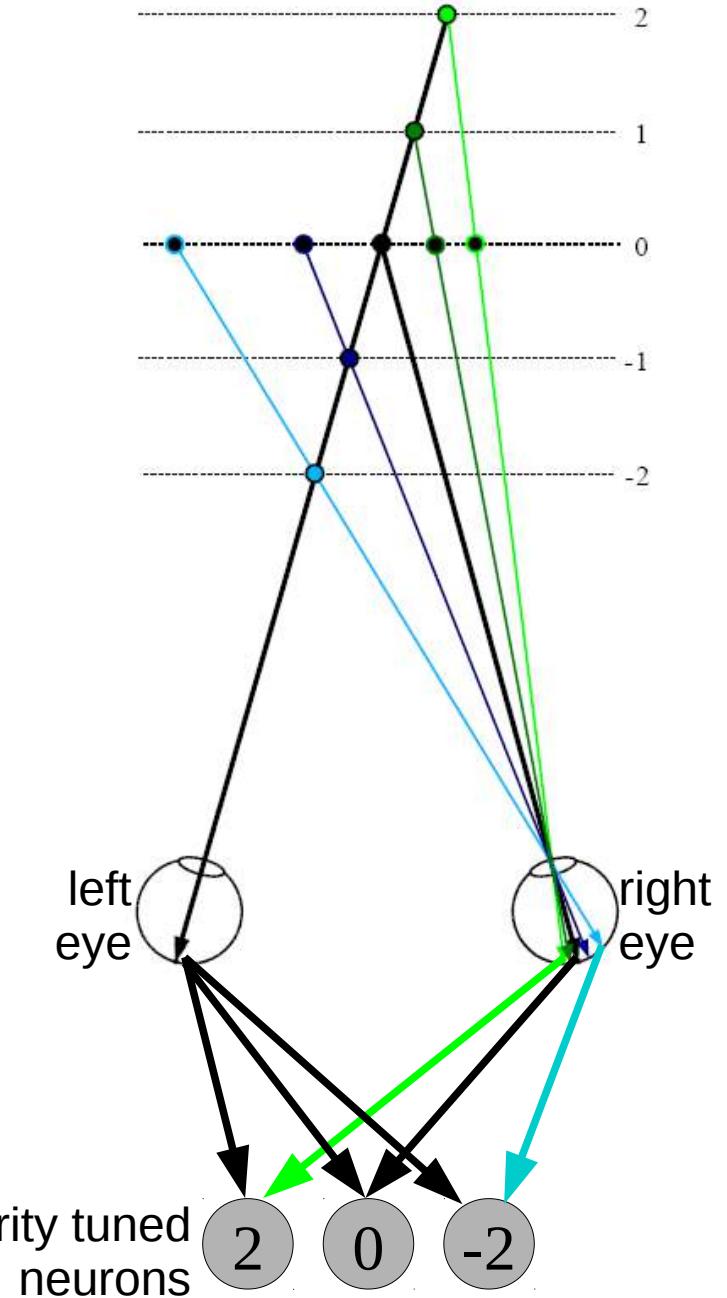
When point F is fixated:

- The images of F fall on the foveas of each retina.
- The images of points on the horopter (e.g. H) fall an equal distance from the foveas of each retina.
- The images of points nearer than the horopter (e.g. N) are displaced outwards (“crossed” disparities)
- The images of points more distant than the horopter (e.g. D) are displaced inwards (“uncrossed” disparities)
- The further the point from the horopter, the greater the displacement.



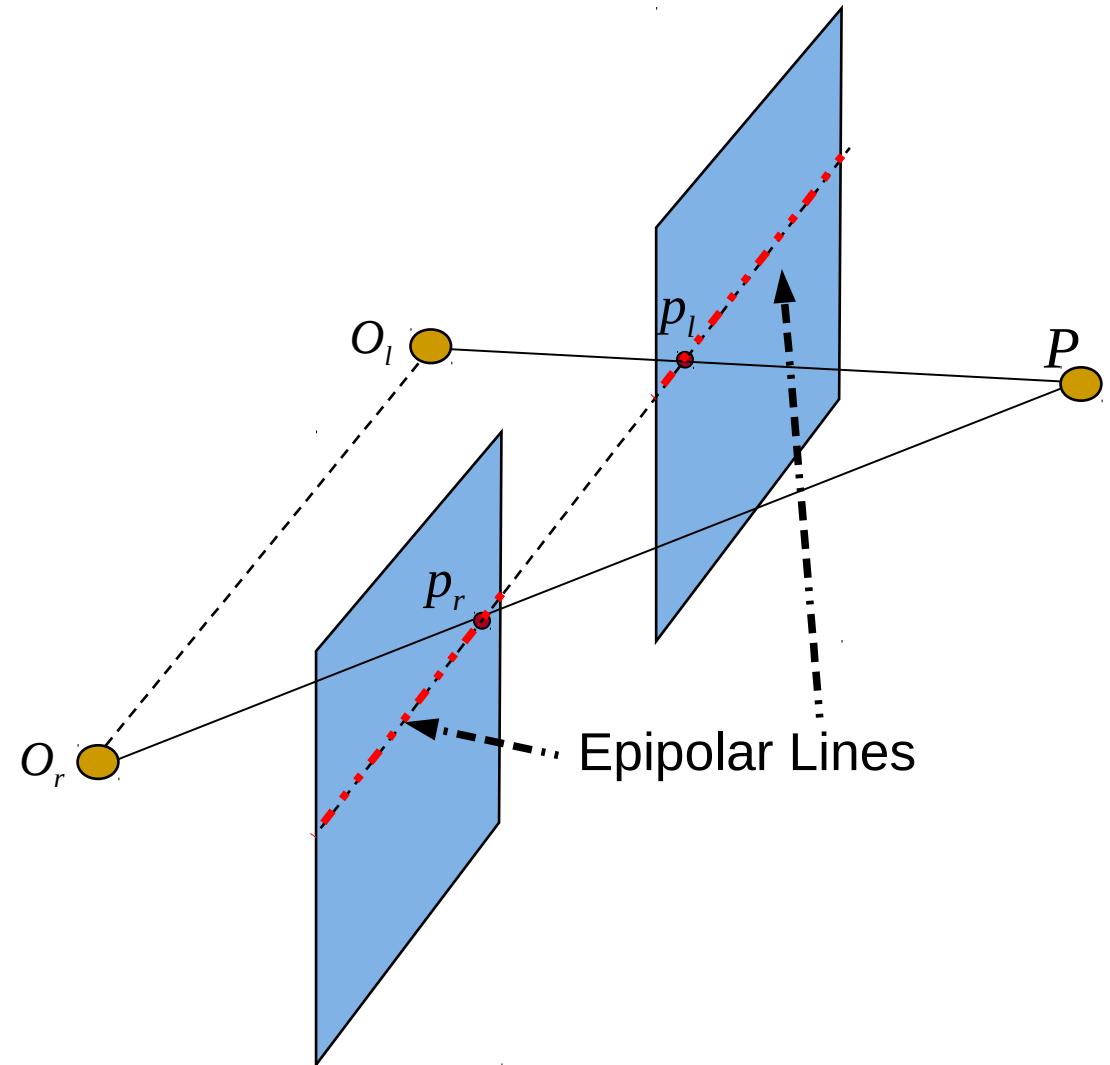
Stereo: non-coplanar eyes

Some cortical neurons are tuned to retinal disparity, and hence, can signal the depth of image points [see lecture 4].



Epipolar geometry

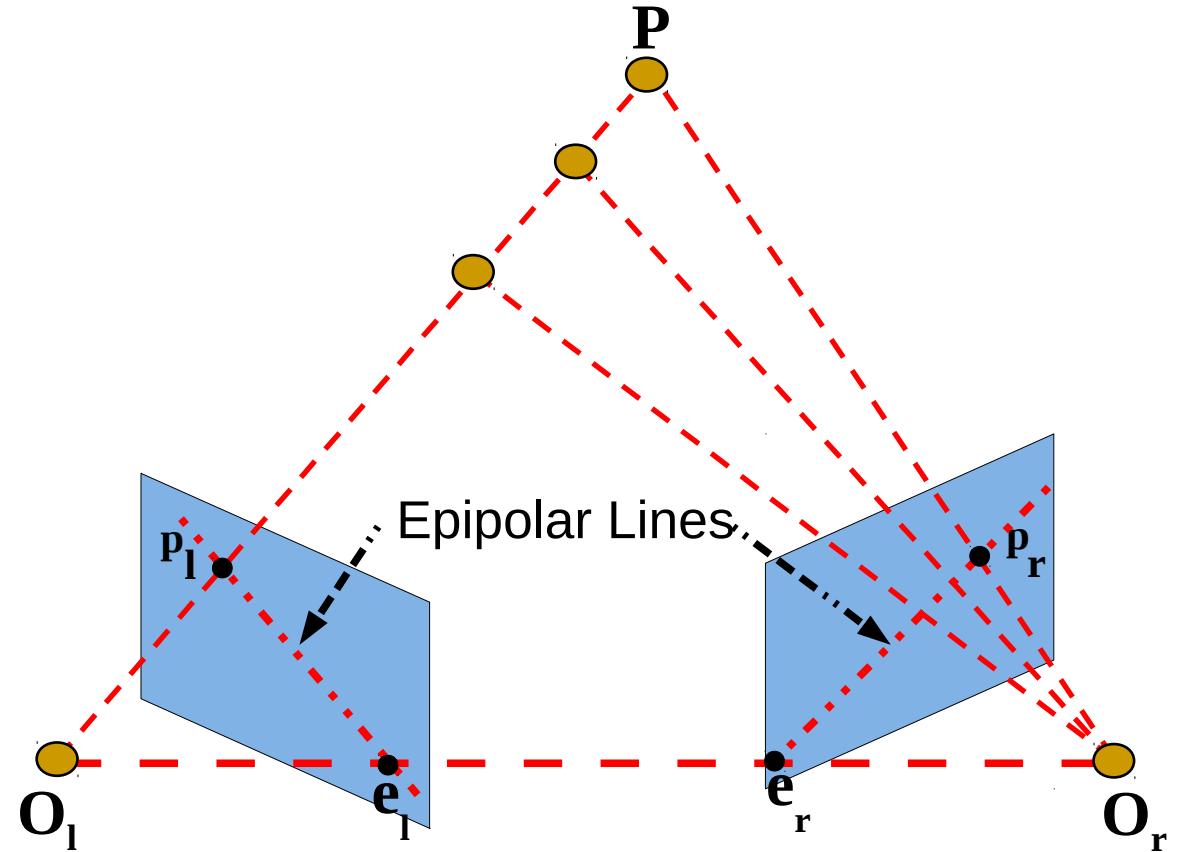
For coplanar cameras, corresponding points in the two images are on corresponding rows of each image.



Epipolar geometry

For non-coplanar cameras, corresponding points still occur on lines, but these are no longer horizontal lines.

A point in the image plane of the left camera corresponds to a line in the image plane of the right camera (and vice versa).



These lines which correspond to points in the other image are called epipolar lines.

2D search for correspondence can be reduced to a 1D search along the “epipolar” line.

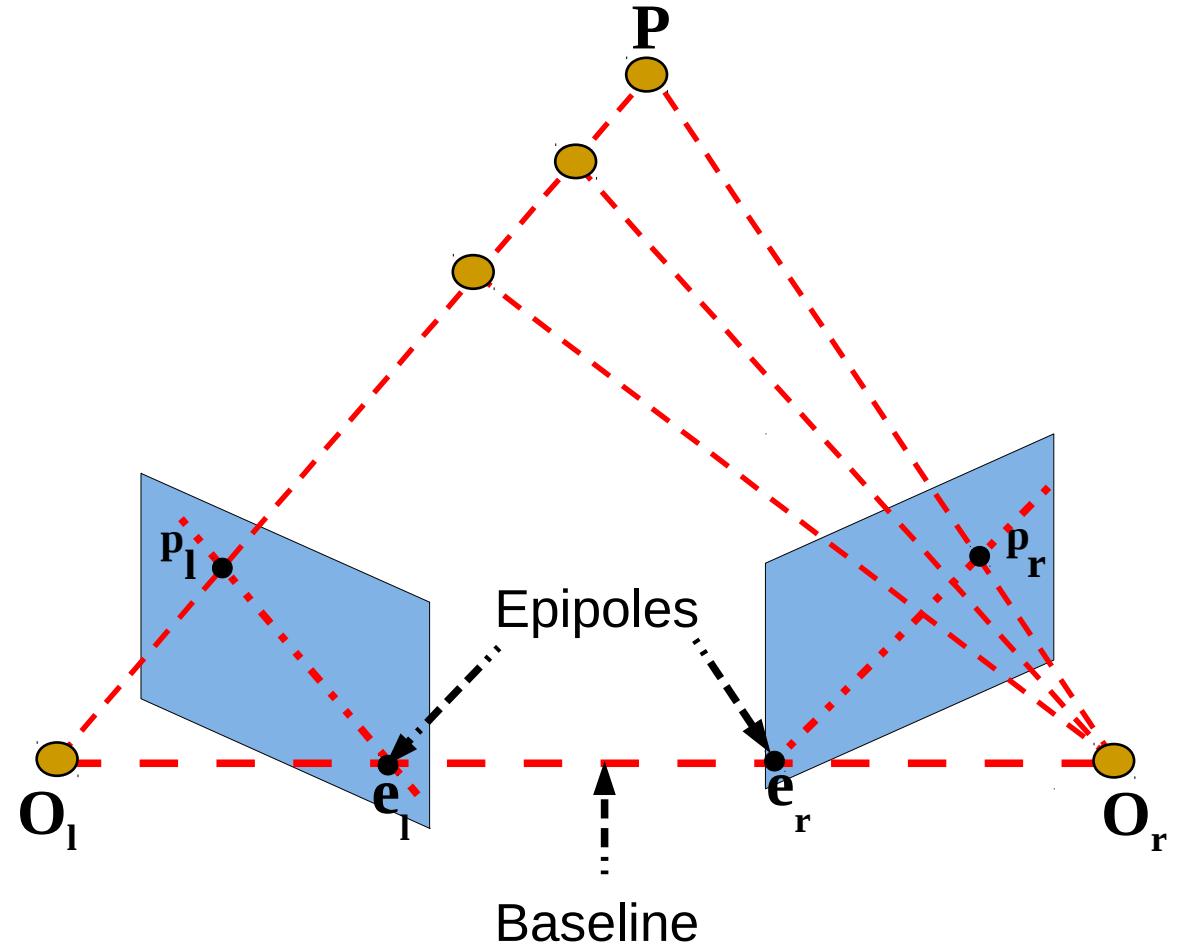
Epipolar geometry: terminology

Baseline: the line through the camera projection centres O_l , O_r

Epipole: projection of the optic centre of one camera on the image plane of the other camera.

The right (left) epipole e_r (e_l) is the image of the left (right) camera projection centre O_l (O_r) in the right (left) image plane.

Equivalently, the epipoles are the intersection of the baseline with each image plane.



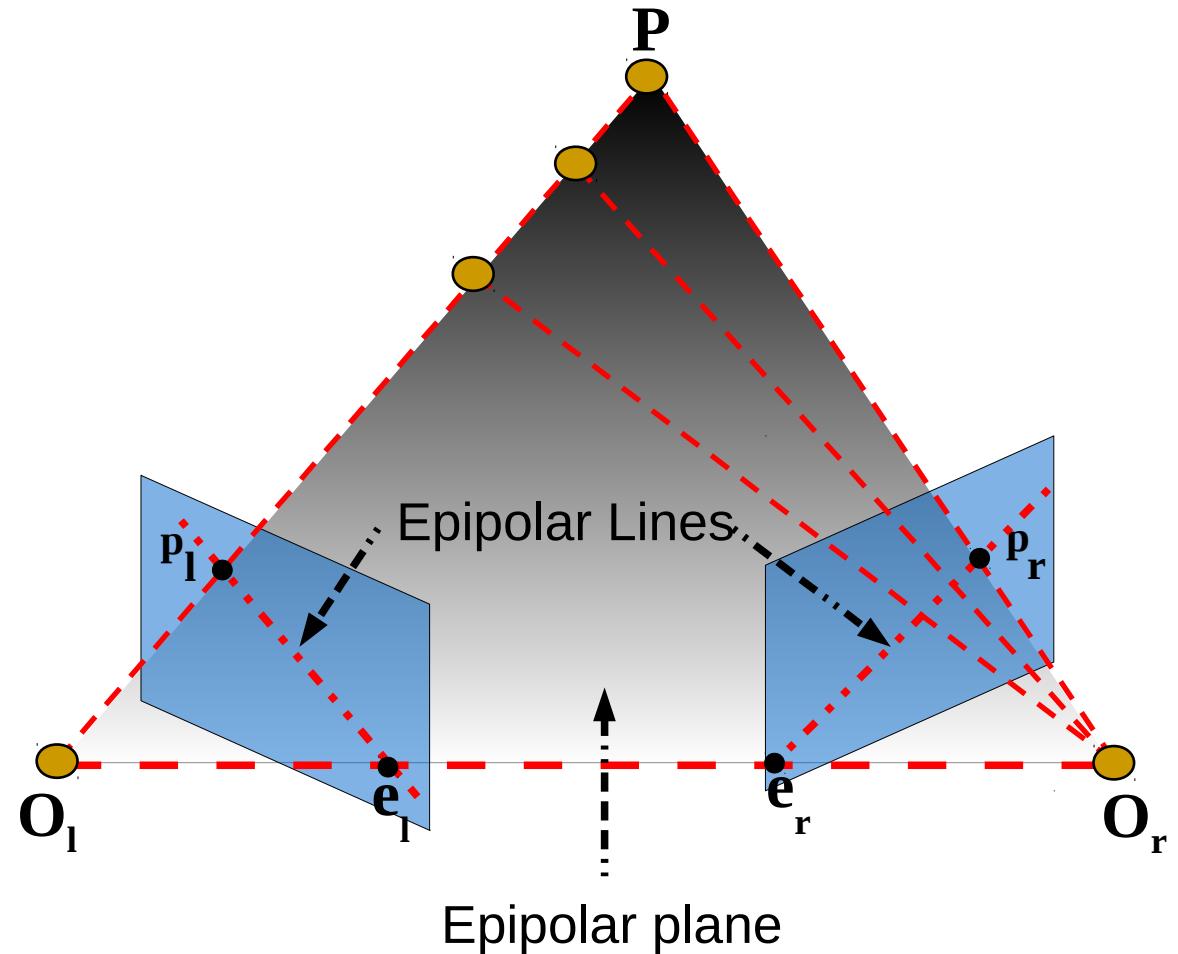
Epipolar geometry: terminology

Epipolar plane: a plane going through a particular 3D point and the optic centres of both cameras.

Epipolar lines: intersection of the epipolar plane (for a particular 3D point) and each image plane.

All epipolar lines in the left image go through e_l , and all epipolar lines in the right image go through e_r .

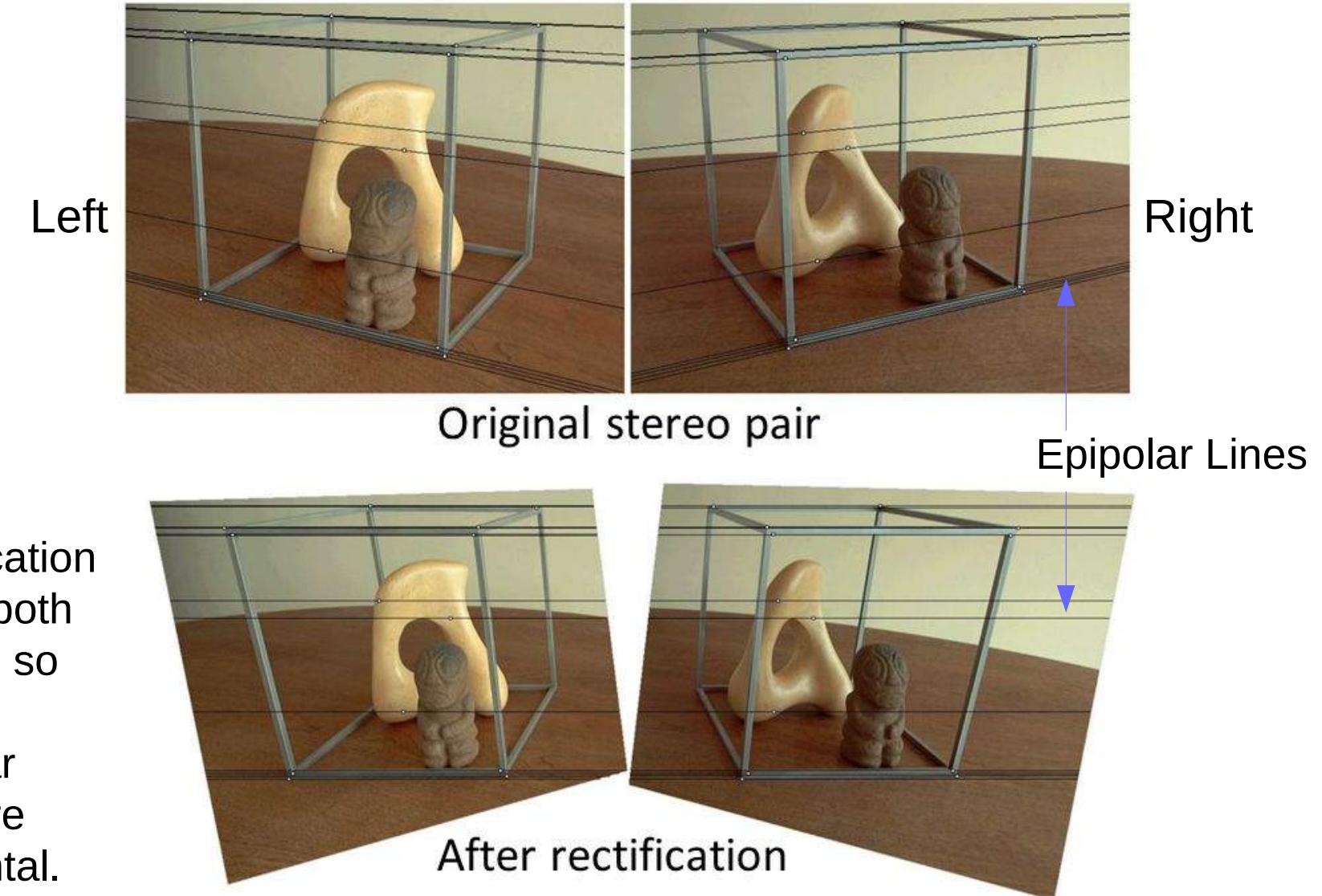
Conjugated epipolar lines: the epipolar lines generated by the same 3D point on the left and right image planes.



Epipolar Constraint: Corresponding points must lie on conjugated epipolar lines.

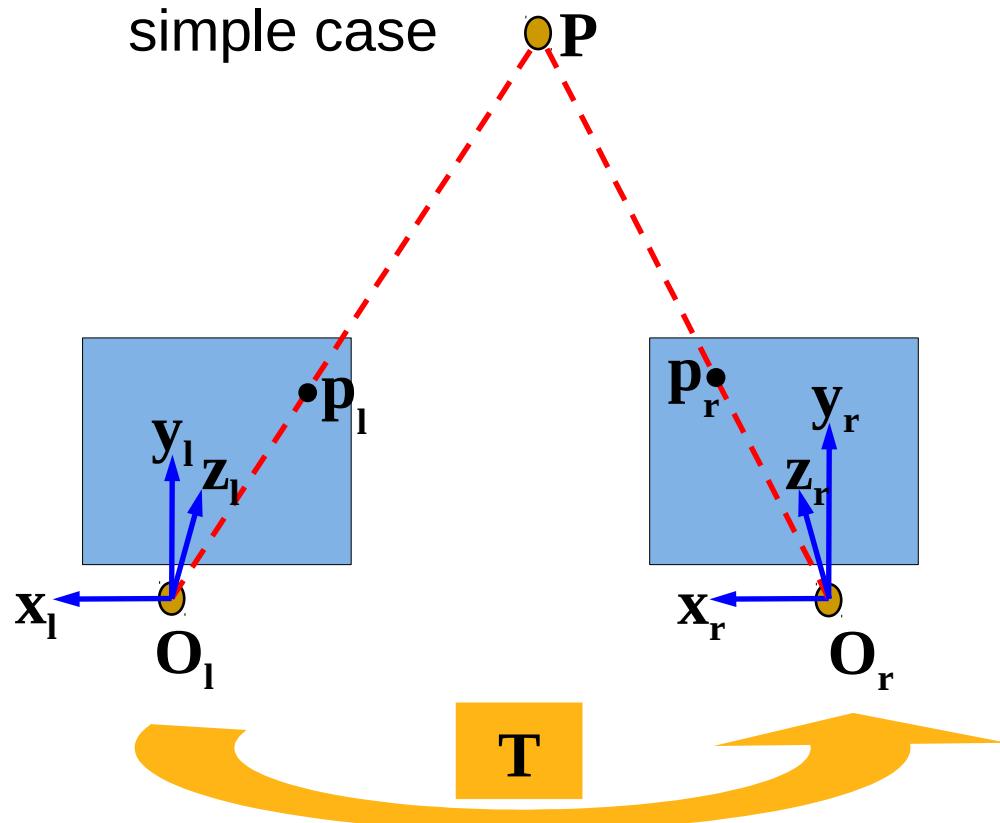
Rectification

Epipolar lines can be made parallel to the rows of the image via a transform called rectification.



Stereo geometry: summary

simple case



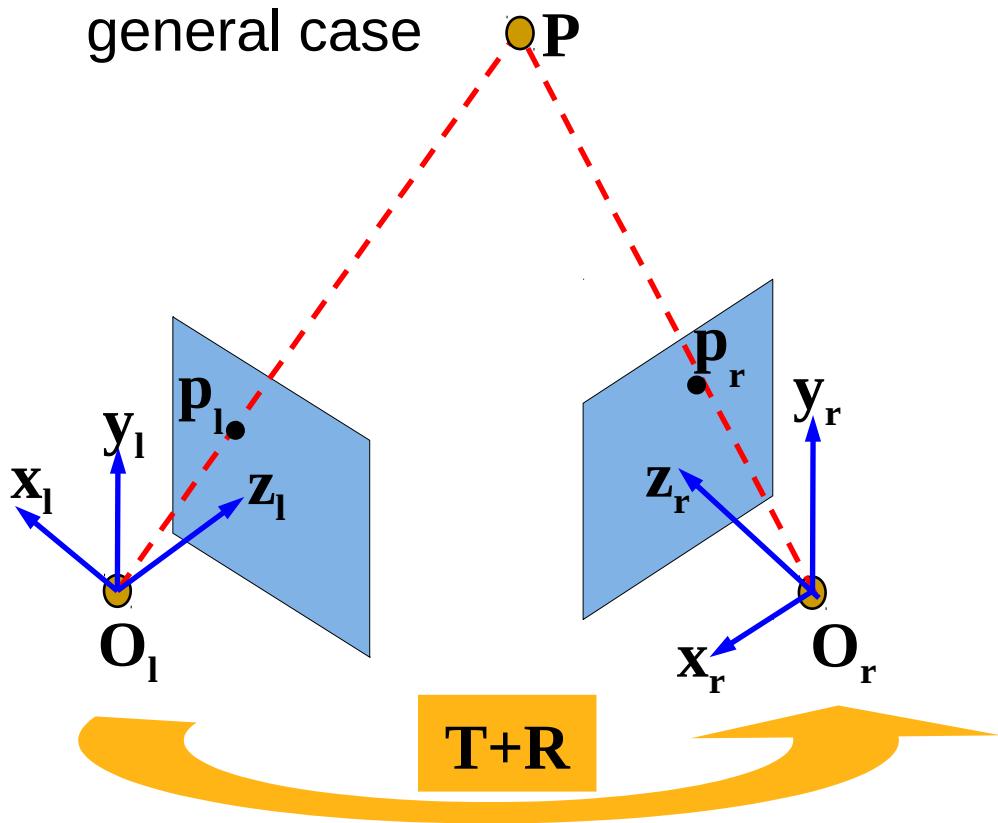
coplanar image planes (cameras translated along x-axis).

epipolar lines are horizontal scan lines

disparity measured in pixels

disparity inversely proportional to depth

general case



noncoplanar image planes (cameras related by a translation and rotation)

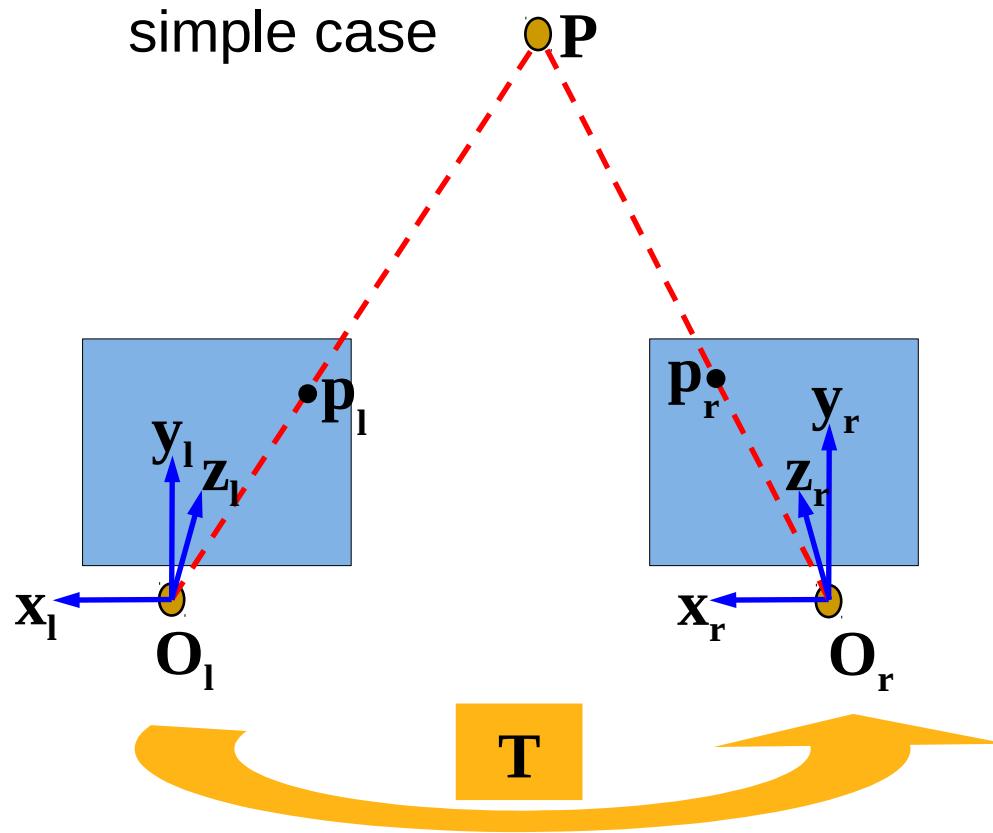
epipolar lines are lines radiating from the epipolar point

disparity measured using angles

disparity increases with distance from horopter

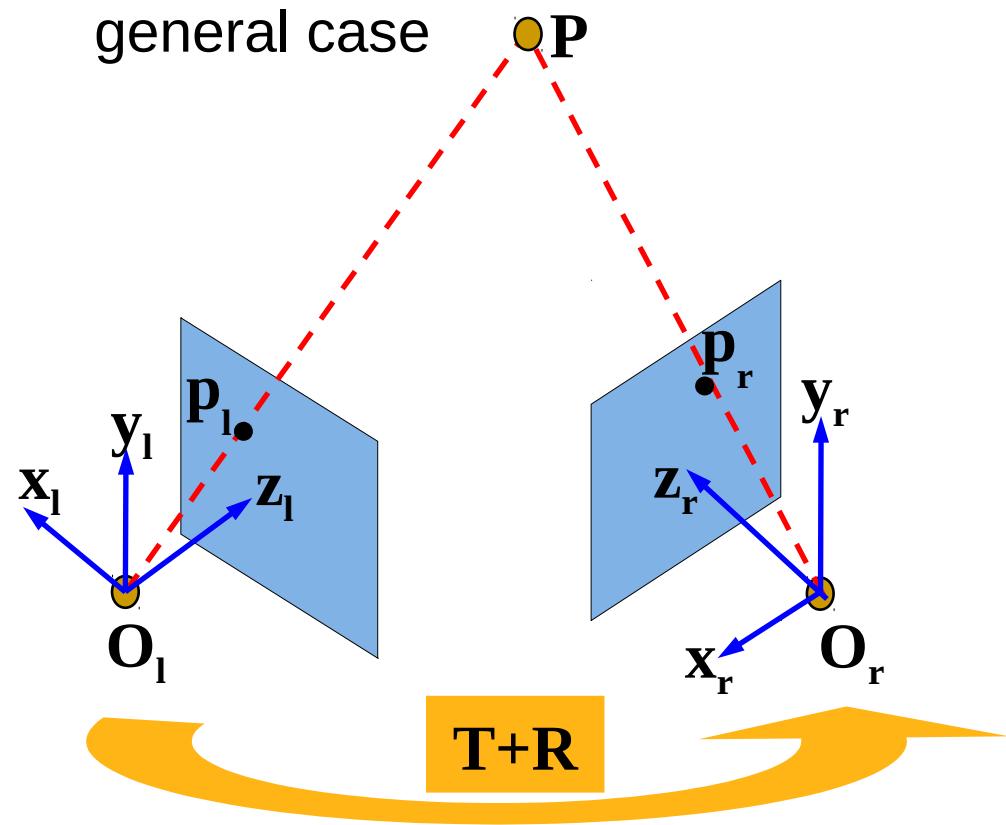
Stereo geometry: summary

simple case



$$\text{depth} = fB/d$$

general case



depth found by, either:

1. triangulation (i.e. determining intersection of corresponding viewlines)
 2. rectification of images and then treating as simple case
- Both require knowledge of intrinsic and extrinsic camera parameters.

Depth

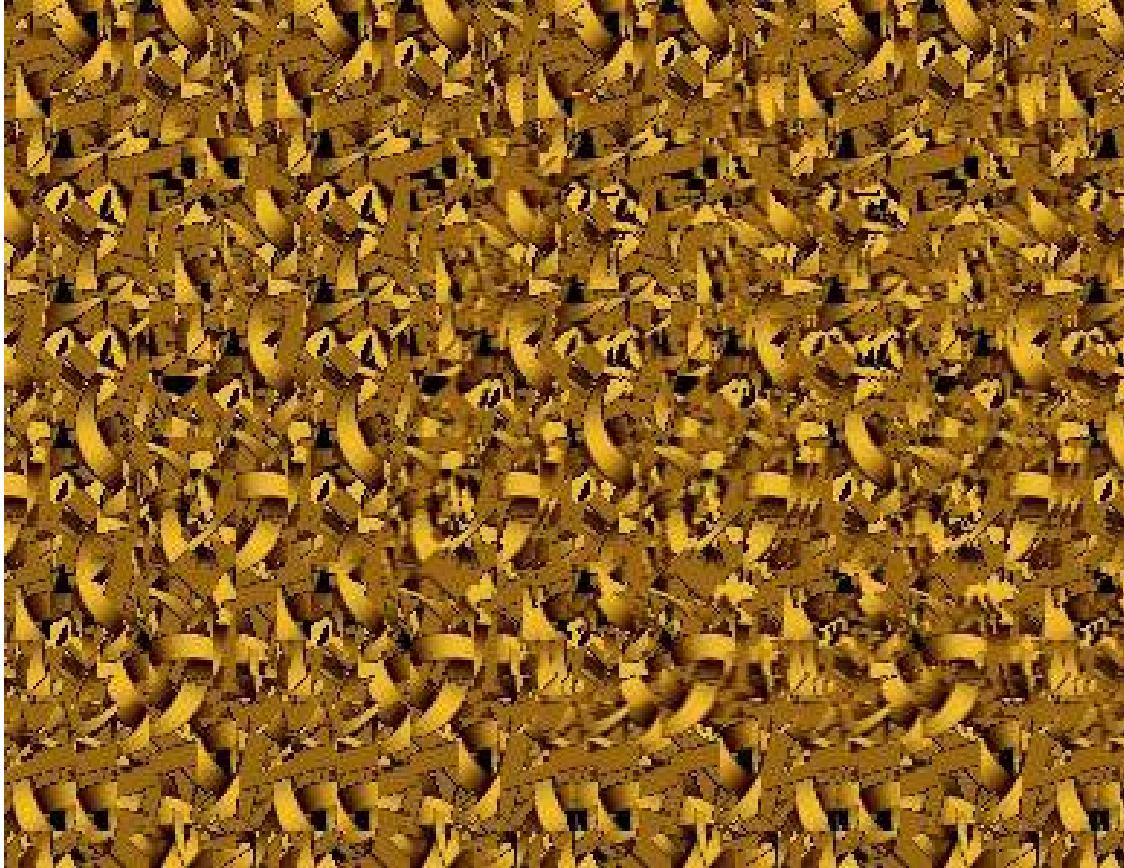
Recovery of depth information is important for:

- Controlling movement
 - mobile robot path planning
 - grasping an object with a robot arm
- 3D reconstruction
 - generating CAD models
 - virtual reality
- Object Recognition
 - depth (between objects) provides a cue for segmentation
 - depth (within an object) provides information about the shape of an object

Stereo is only one source of information about depth...

Cues to depth: Binocular cues

Disparity: difference in location of corresponding points in a stereo pair of images.



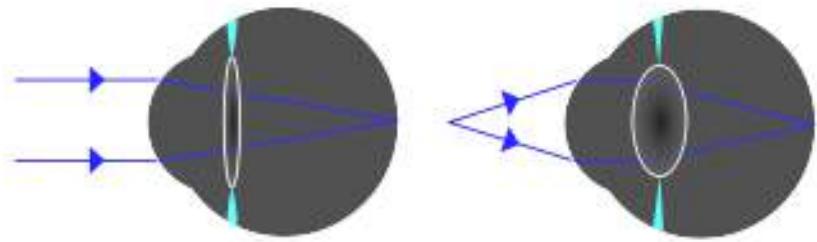
an autostereogram

Stereo vision is important (try catching a ball with one eye closed), and is sufficient (random dot stereograms and autostereograms demonstrate this as all other cues to depth have been removed).

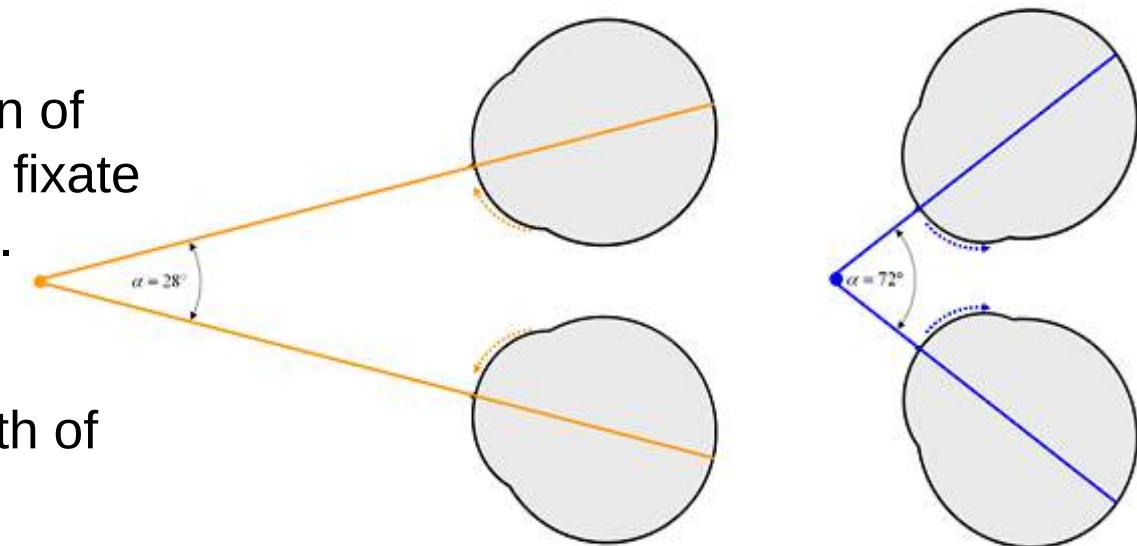
However, disparity is only one of many depth cues.

Cues to depth: Oculomotor cues

Accommodation: the shape of the lens (and hence its refractive power) can be modified by muscles in the eye to bring objects at different depths into focus. Hence, the state of these muscles provides information about the depth of the object being observed.



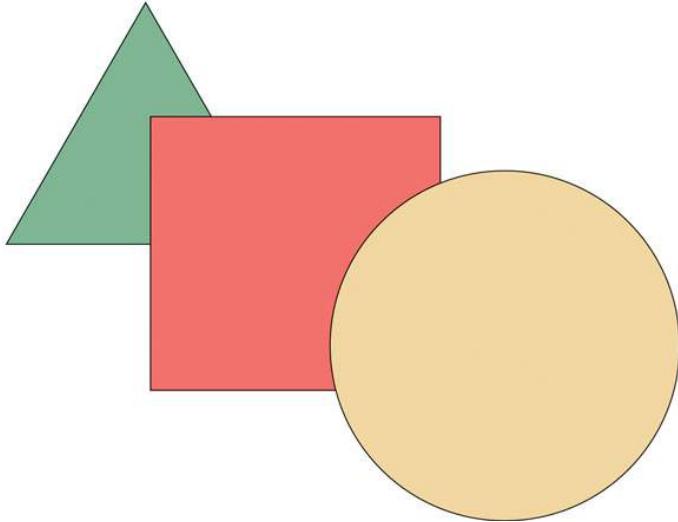
Convergence: the rotation of eyes/cameras can vary to fixate objects at different depths. Hence, the angle of convergence provides information about the depth of the object being fixated.



Cues to depth: Monocular cues

i.e. cues that provide depth information with one eye closed

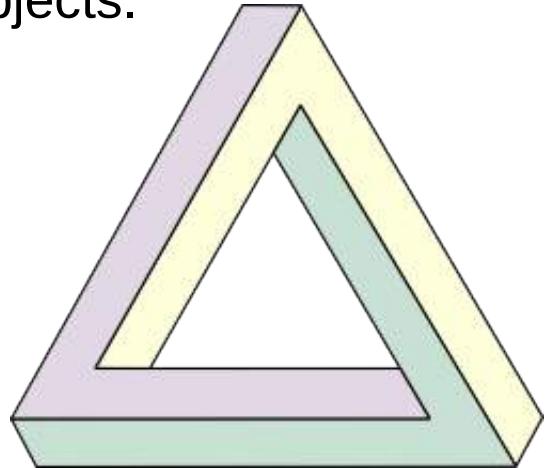
Interposition: when the view of one object is interrupted by the presence of another object, we use this pattern of occlusion to determine the relative depth of the objects. The near object is perceived as “interposed” between the far object and the observer.



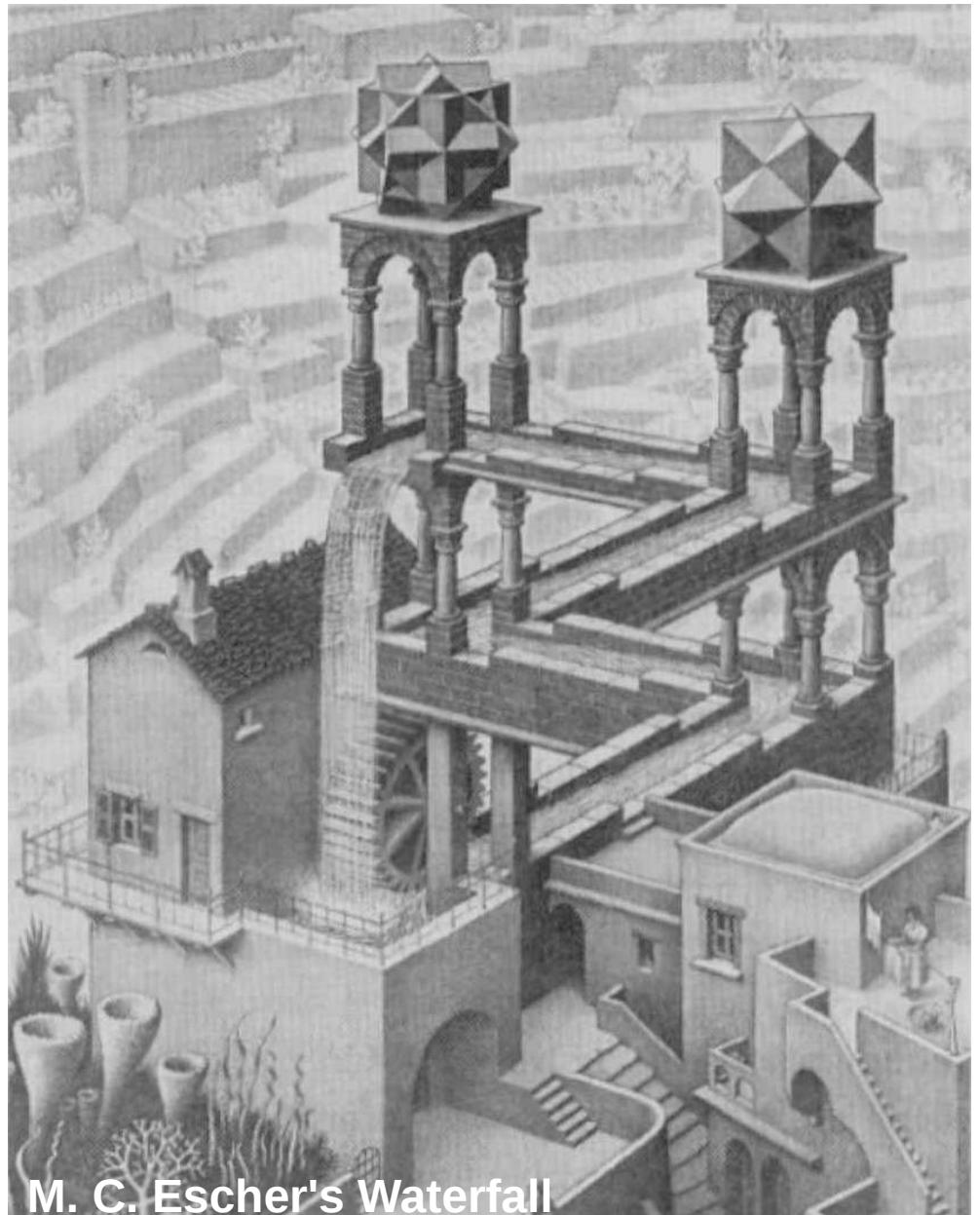
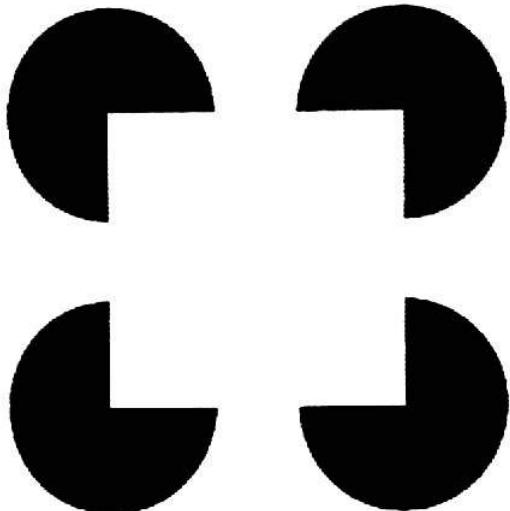
Cues to depth: Monocular cues

Interposition: manipulating interposition depth cues can produce impossible objects, or illusory objects.

Penrose triangle



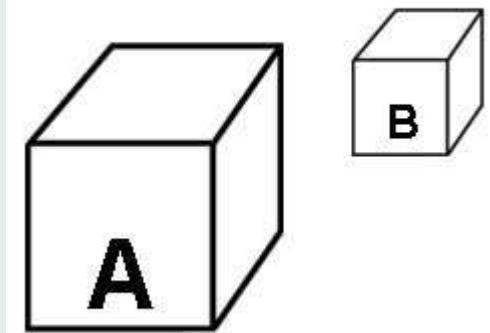
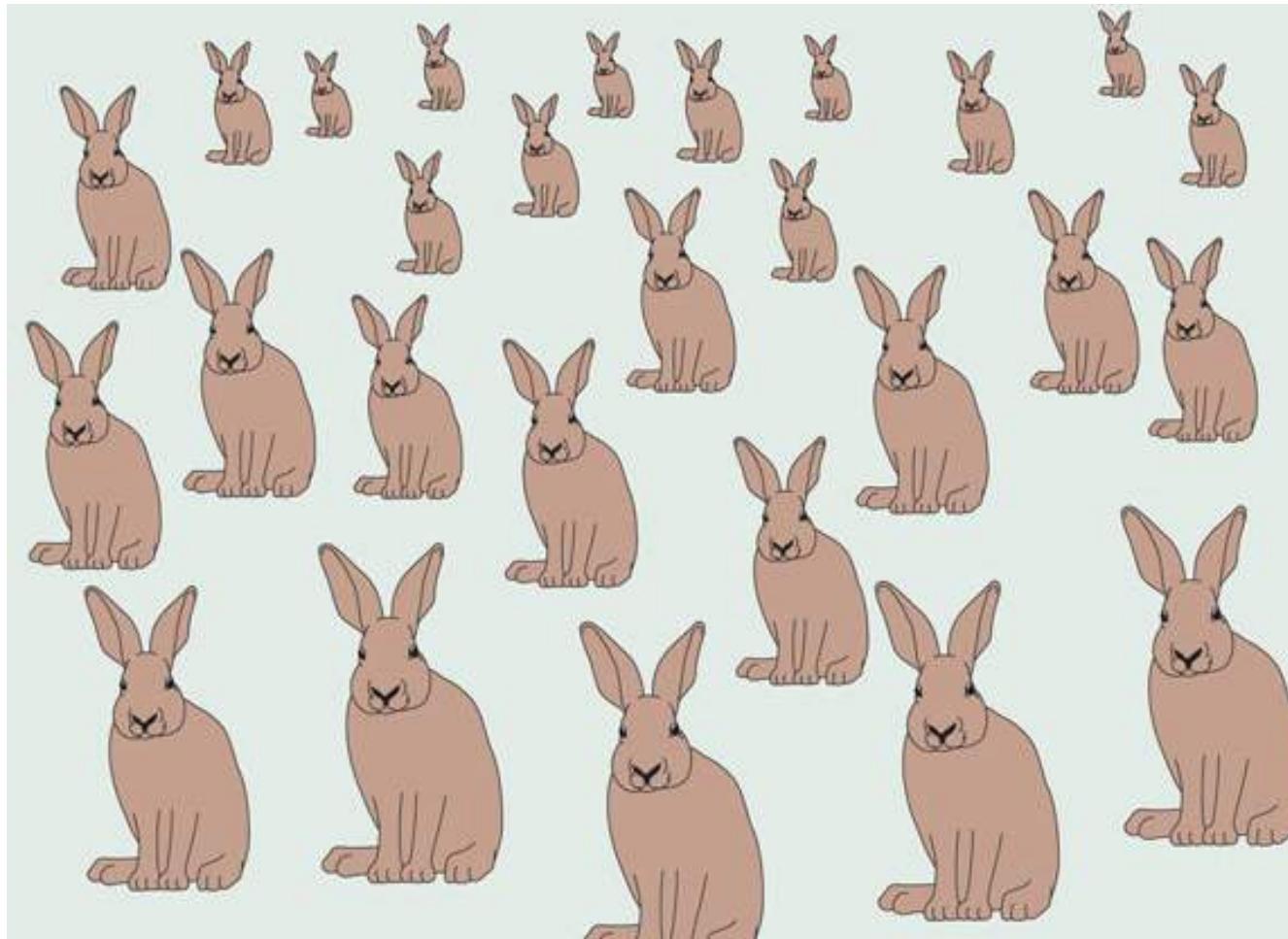
Kanizsa square



M. C. Escher's Waterfall

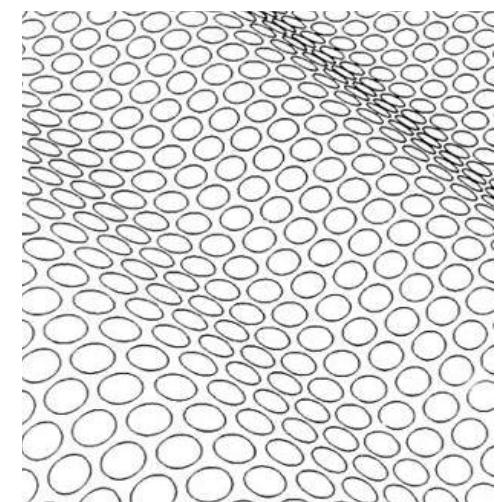
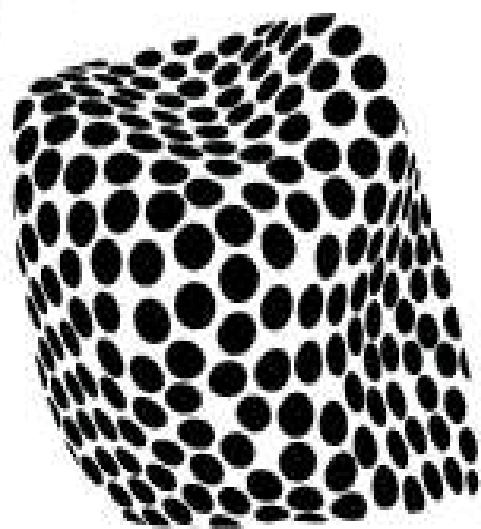
Cues to depth: Monocular cues

Size familiarity: distant objects necessarily produce a smaller image than nearby objects of the same size. The larger of two identical objects tends to be perceived as closer than the smaller one.



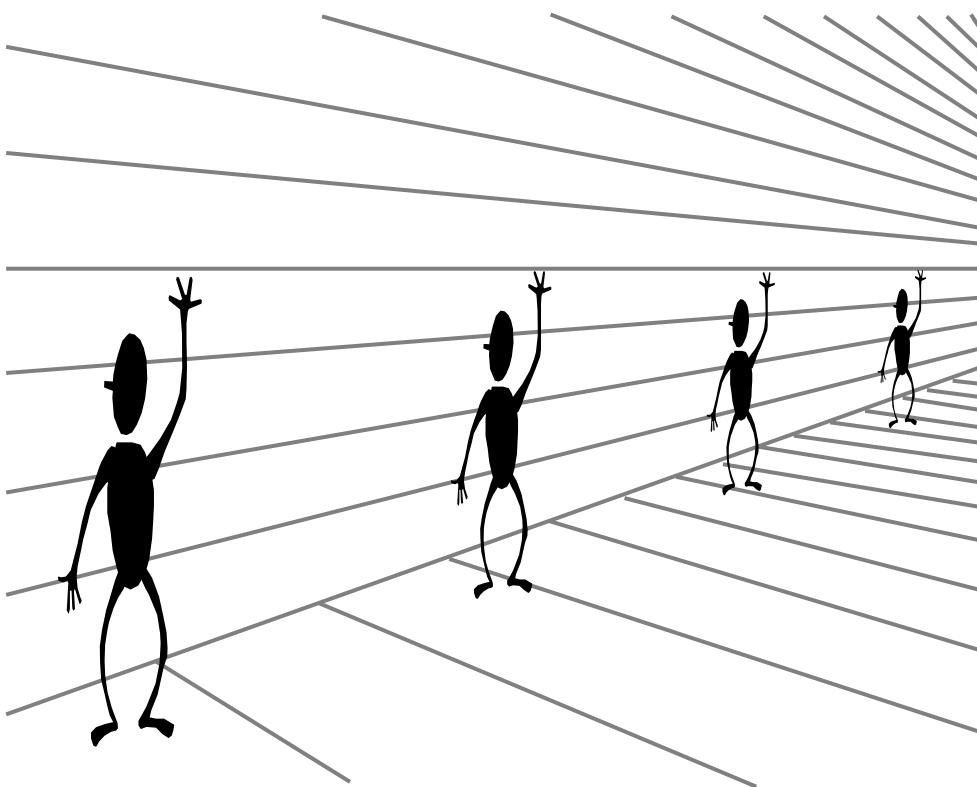
Cues to depth: Monocular cues

Texture gradients: for uniformly textured surfaces, the texture elements get smaller and more closely spaced with distance.



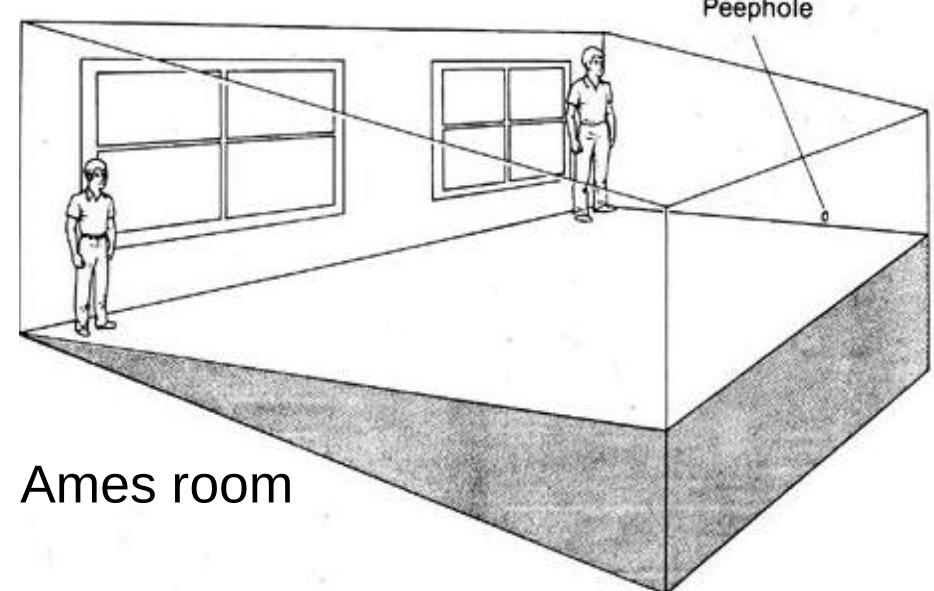
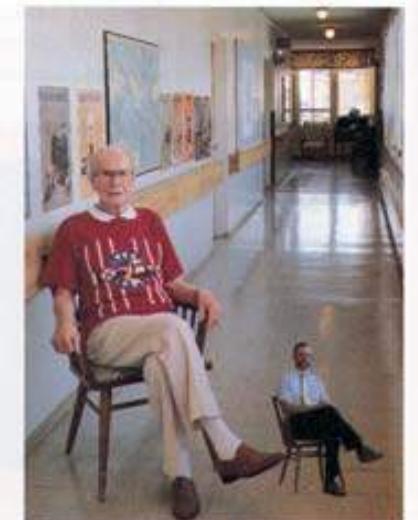
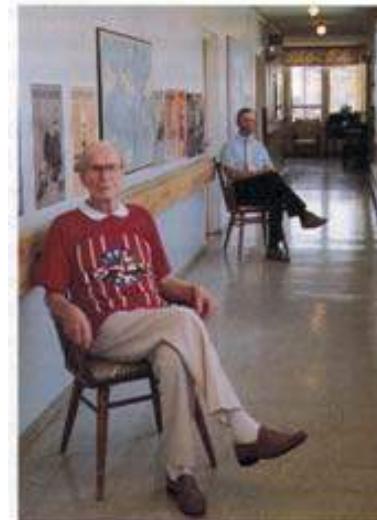
Cues to depth: Monocular cues

Linear perspective: the property of parallel lines converging at infinity provides a cue to depth (and size).



Cues to depth: Monocular cues

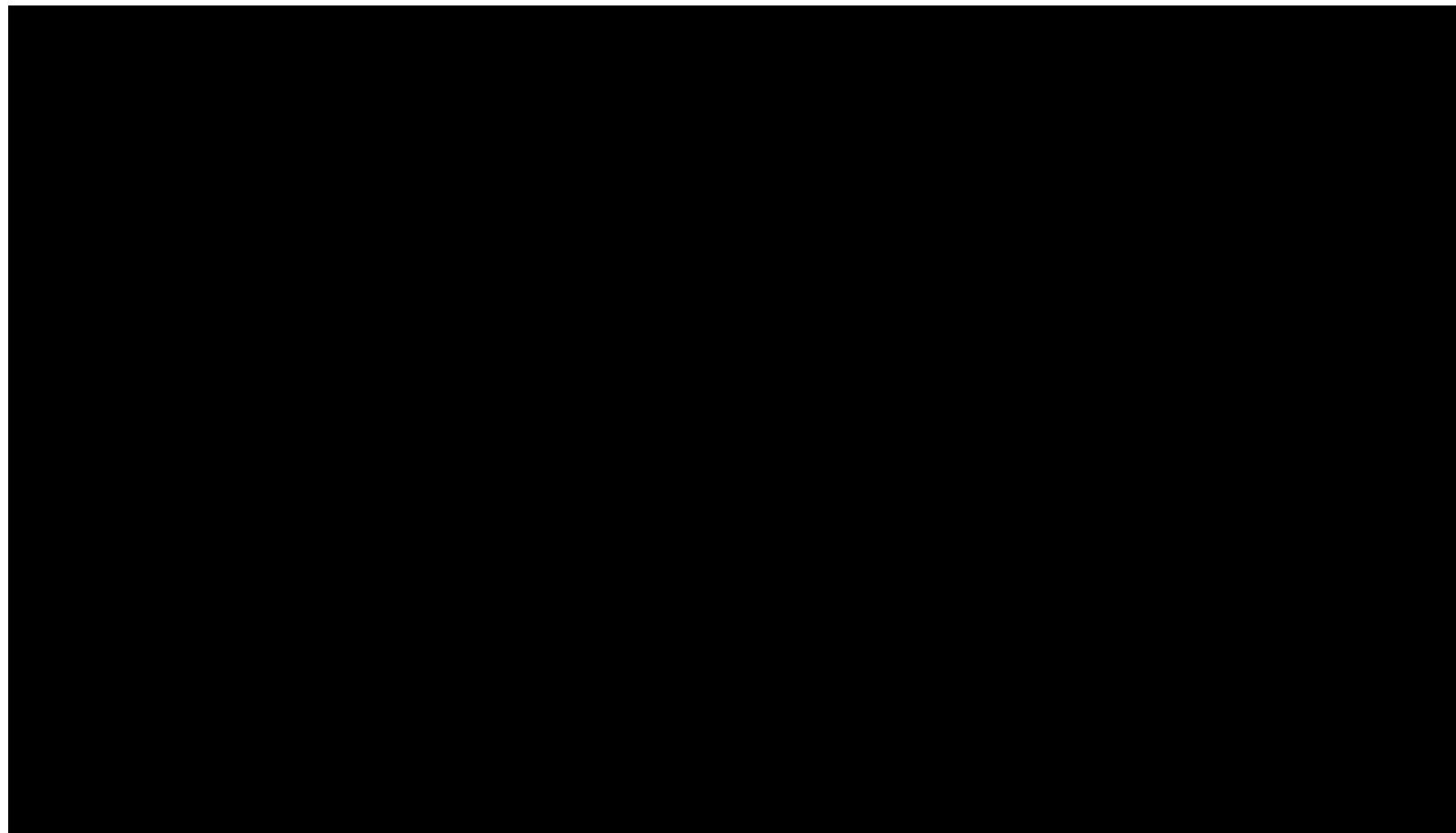
Linear perspective: manipulating perspective can produce unusual perceptions (overcoming size familiarity).



Ames room

Cues to depth: Monocular cues

Linear perspective: manipulating perspective can produce unusual perceptions (overcoming prior beliefs about gravity!).



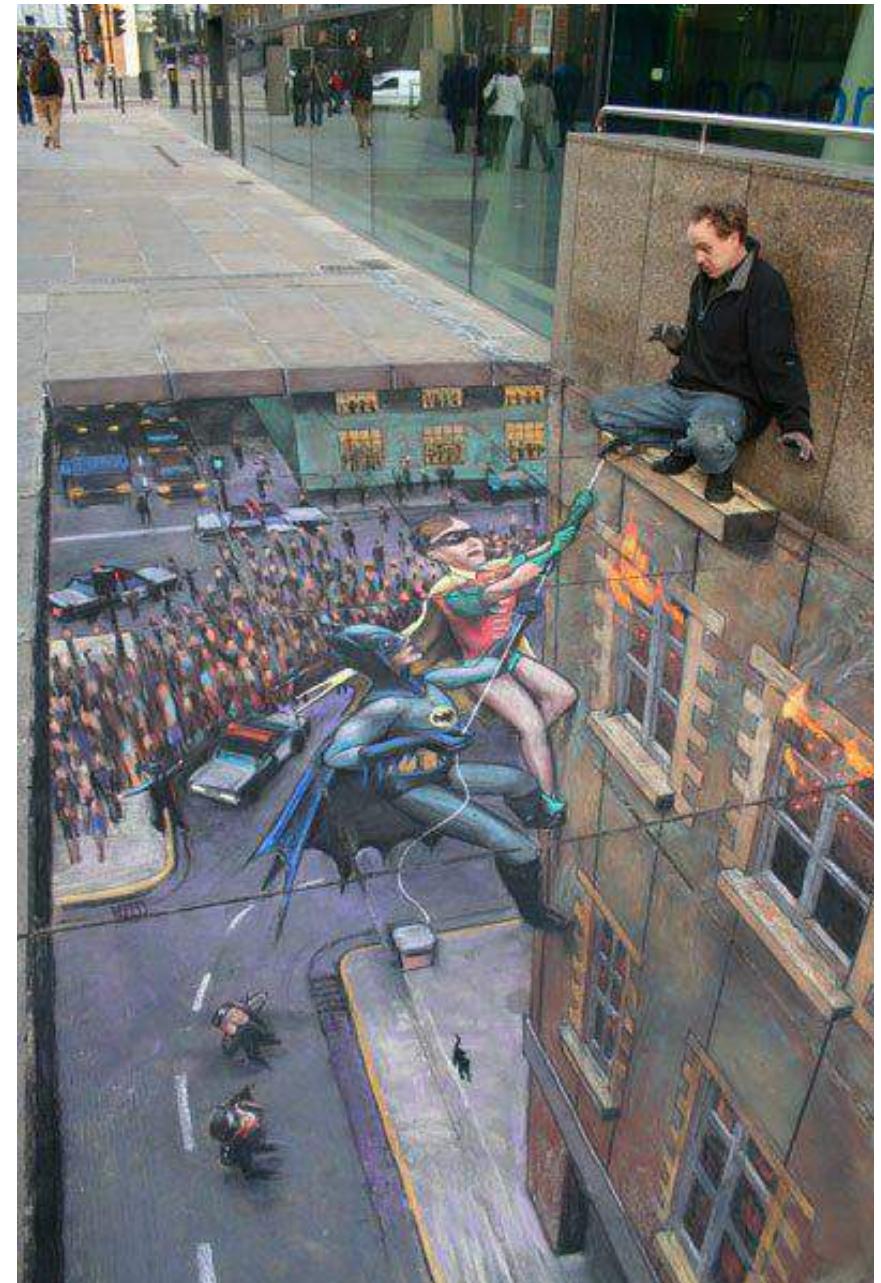
Cues to depth: Monocular cues

Linear perspective: manipulating perspective can produce a strong perception of 3D structure, e.g. Trompe L'oeil art.



Cues to depth: Monocular cues

Linear perspective: manipulating perspective can produce a strong perception of 3D structure, e.g. Trompe L'oeil art.



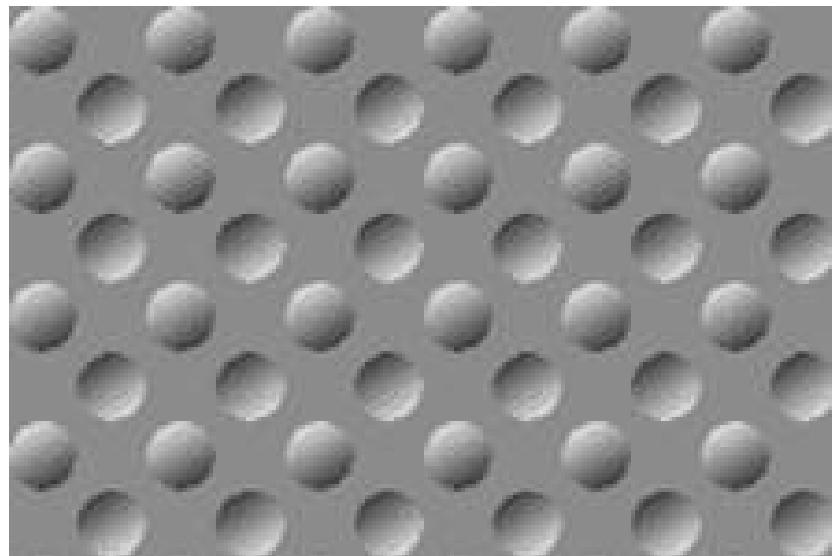
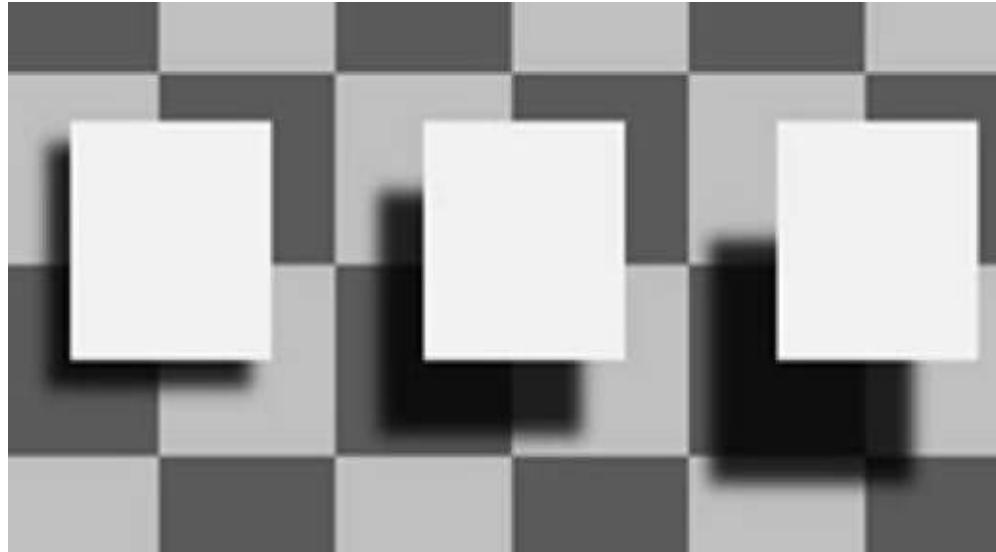
Cues to depth: Monocular cues

Aerial perspective: due to the scattering of light by particles in the atmosphere, distant objects look fuzzier and have lower luminance contrast and colour saturation.



Cues to depth: Monocular cues

Shading: the distribution of light and shadow on objects provides a cue for depth (the brain assumes, usually, that light comes from above).



Cues to depth: Monocular cues

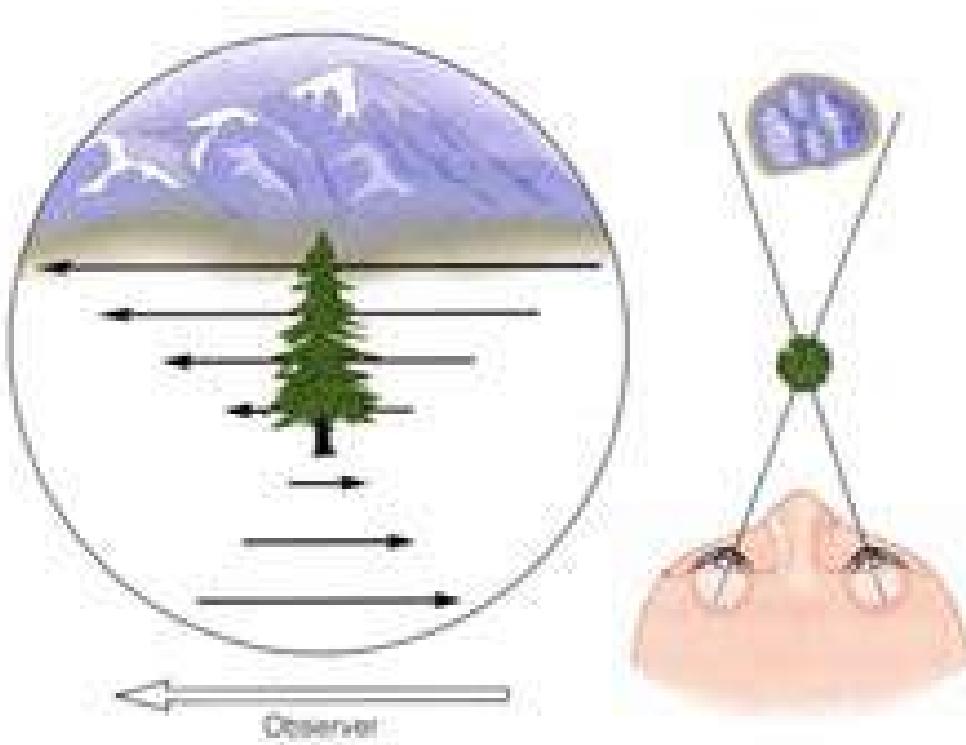
Shading: the distribution of light and shadow on objects provides a cue for depth (the brain assumes, usually, that light comes from above).

Hollow face illusion
shading fails to
recover depth when
pitted against
familiarity



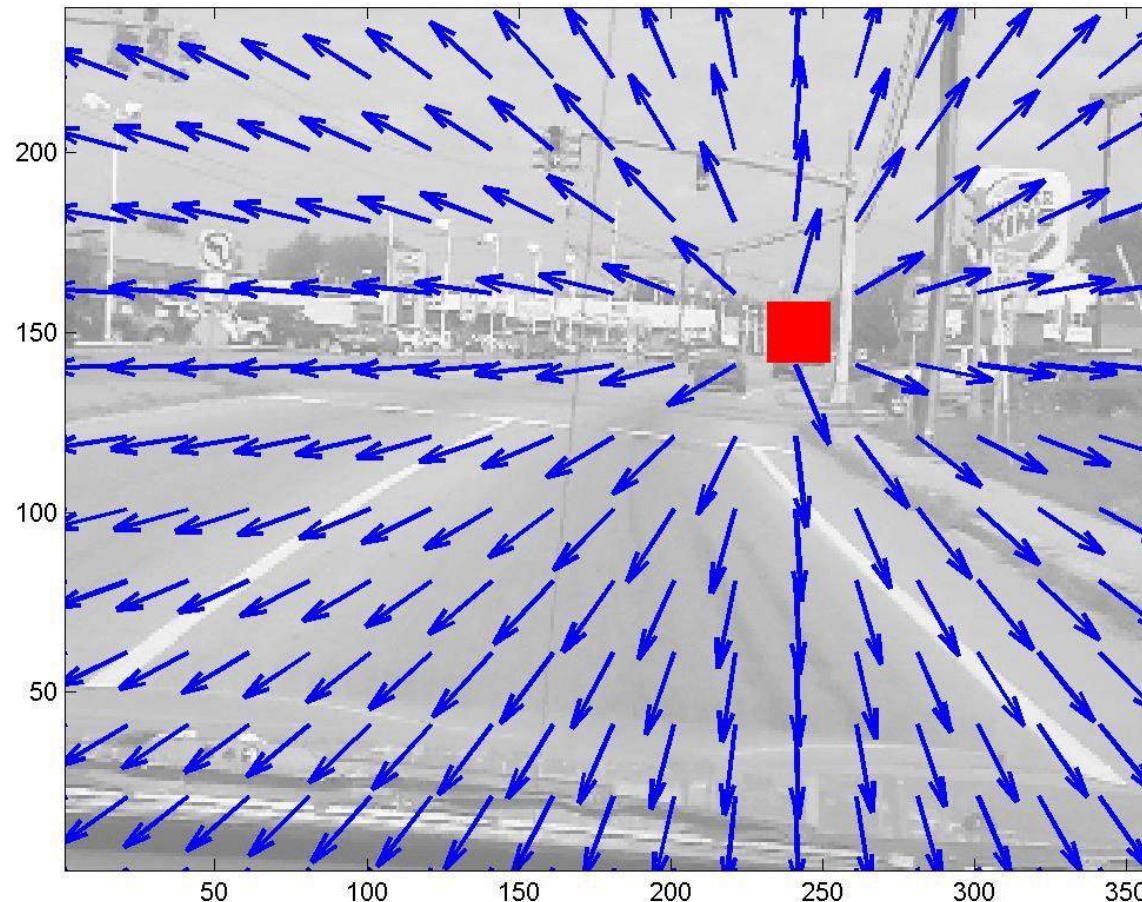
Cues to depth: Motion induced cues

Motion parallax: speed and direction of image motion induced by movement of the camera/eye varies with depth. Objects closer than the fixation point appear to move in a direction opposite to the observer, while objects further away appear to move in the same direction.



Cues to depth: Motion induced cues

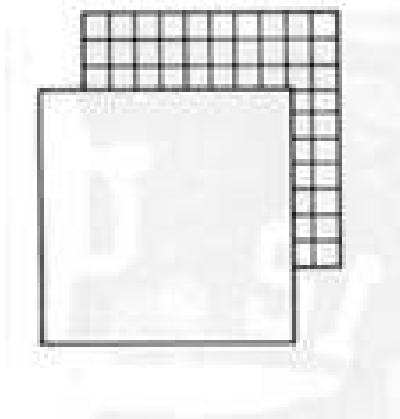
Optic Flow: As a camera moves forward or backward, the pattern of stimulation across the entire visual field changes, producing a pattern of expanding or contracting “optic flow”. Points closer to the camera move more quickly across the image plane



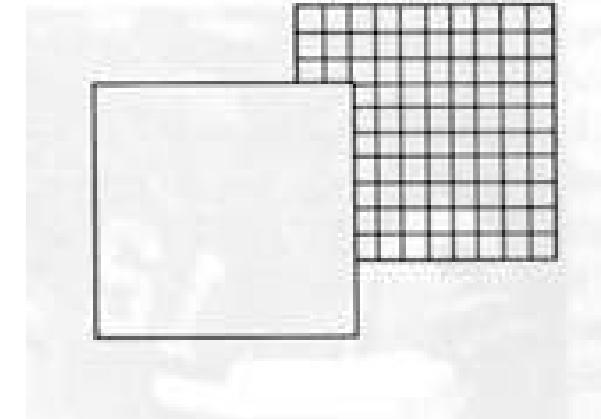
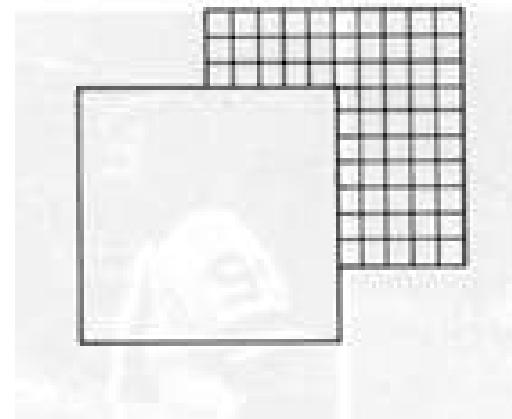
Cues to depth: Motion induced cues

Accretion and deletion : parts of an object can appear or disappear when an observer moves relative to two surfaces that are at different depths.

Motion to left causes far surface to become occluded by near surface



Motion to right causes far surface to become uncovered by near surface



Cues to depth: Motion induced cues

Structure from motion (kinetic depth): movement of an object or of the camera can induce the perception of 3D structure.



Depth is conveyed solely by the variation in velocity and direction of each dot: no depth is seen in static image.



Summary: Stereo Vision

Aim: to infer depth from 2 or more images taken from different viewpoints. Two sub-problems:

1. Correspondence. Determining which points in the left and right images are projections of the same point in the scene. Constraints:

- epipolar constraint (corresponding points lie along the epipolar line)
- maximum disparity (extent of search reduced by knowledge of baseline and minimum depth)
- continuity (disparity varies smoothly assuming continuous surfaces)
- uniqueness (one-to-one matches assuming surfaces are approximately parallel to image plane and no occlusion)
- ordering (points occur in same order in each image assuming no occlusion)

Summary: Stereo Vision

2. Reconstruction. Given the correspondence between points, and camera parameters, calculate the depth (z).

Depth related to disparity (d).

For coplanar cameras: $d = x'_L - x'_R$

$$z = f \frac{B}{d}$$

For non coplanar cameras: $d = \alpha_L - \alpha_R$

$d > 0$ outside of the horopter

$d < 0$ inside the horopter

Summary: Depth

Binocular

Disparity

Oculomotor

Accommodation (shape of lens in eye)

Convergence (angle of rotation of cameras/eyes)

Monocular

Interposition (occlusion of one object/part of object by another)

Size familiarity (the smaller the object the greater its depth)

Texture gradients (texture become smaller and denser with depth)

Linear perspective (convergence of lines at vanishing points)

Aerial perspective (reduction in contrast and saturation with depth)

Shading (light and shadow)

Motion

Motion parallax (depth related to image motion caused by camera motion)

Optic Flow (depth related to speed of image expansion/contraction)

Accretion and deletion (changes in occlusion due to camera movement)

Structure from motion (depth perception due to object motion)

Computer Vision (7CCSMCVI / 6CCS3COV)

Recap

- **Image formation**
- **Low-level vision**
- **Mid-level vision**
 - grouping and segmentation of image elements
 - **Biological**
 - bottom-up influences (Gestalt cues)
 - top-down influences (knowledge, expectation, etc.)
 - **Artificial**
 - Thresholding, Region-based, Clustering, Fitting
 - **Multi-View Vision**
 - Stereo and Depth
 - Video and Motion
- **High-level vision**

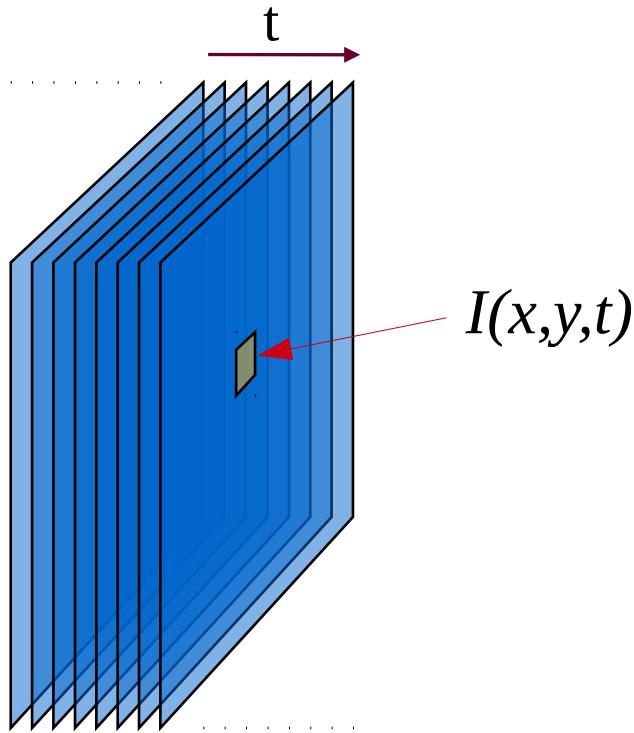
← Today

Today

- Optic flow
 - motion fields
 - measurement
 - » correspondence problem
 - » aperture problem
 - applications
 - » depth
 - » time-to-collision
- Tracking
- Segmentation
 - image differencing
 - background subtraction

Video

Video is a series of N images, or frames, acquired at discrete time instants $t_k = t_0 + k\Delta t$, where Δt is a fixed time interval and $k=0,1,\dots,N-1$



In static images

The intensity of a pixel can be seen as a function of its spatial coordinates (x,y) :
i.e. an image is $I(x,y)$

In video

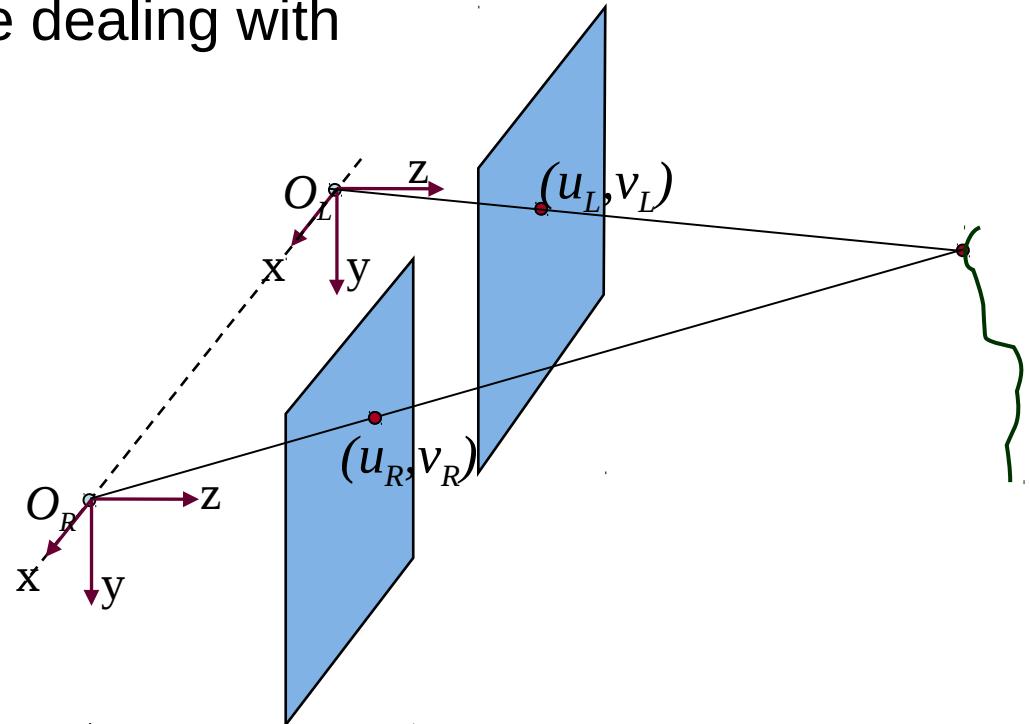
The intensity of a pixel can be seen as a function of its spatial coordinates (x,y) and time (t) :
i.e. a video is $I(x,y,t)$

Video and Stereo

Similar to stereo in that we are dealing with more than one image

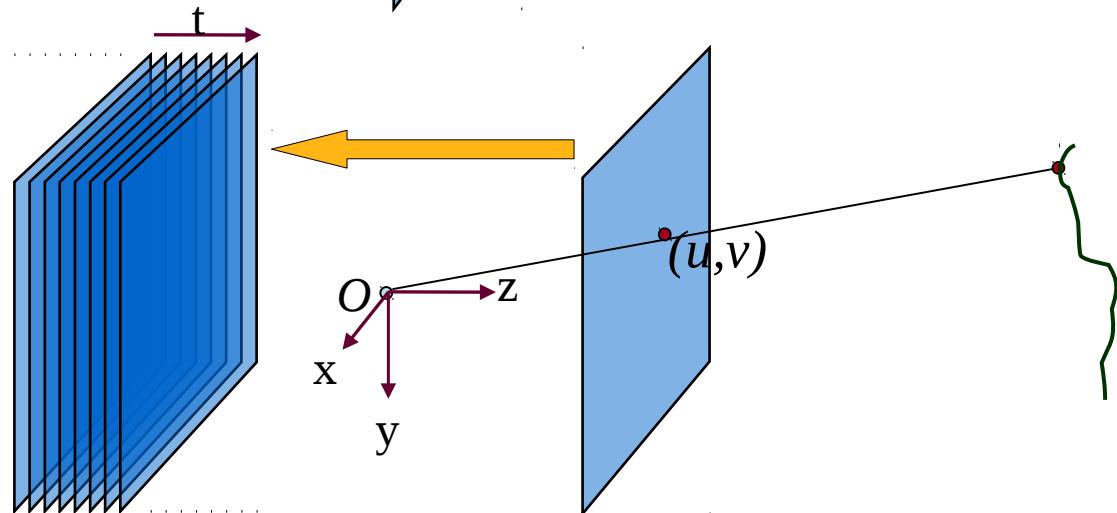
Stereo

- » multiple cameras
- » one time
(images taken simultaneously)



Video

- » one camera
- » multiple times
(images taken at different times)

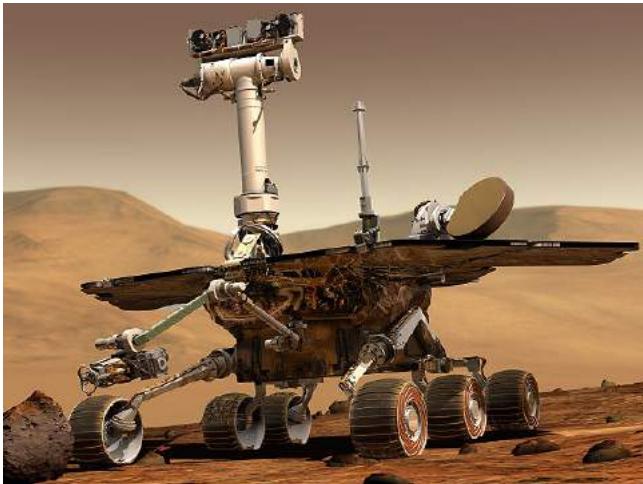


Video

Enables:

- inference of 3D structure (as with stereo).
- segmentation of objects from background without recovery of depth (unlike stereo)
- inference of self and object motion (unlike stereo)
 - essential for some applications, e.g.:

robot navigation



driver assistance



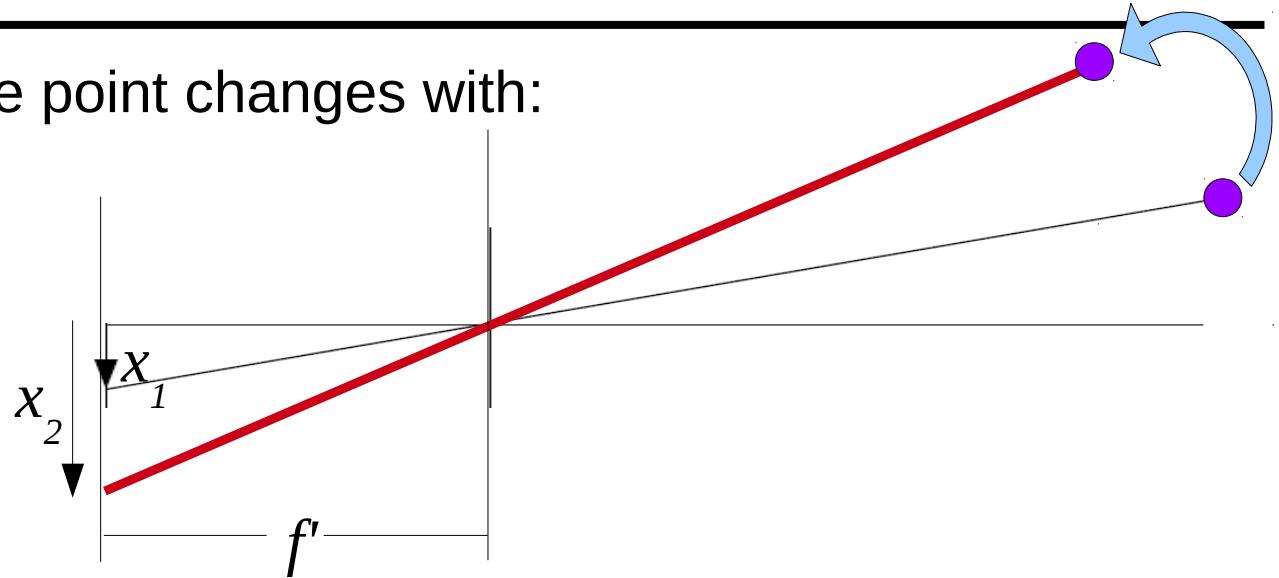
surveillance



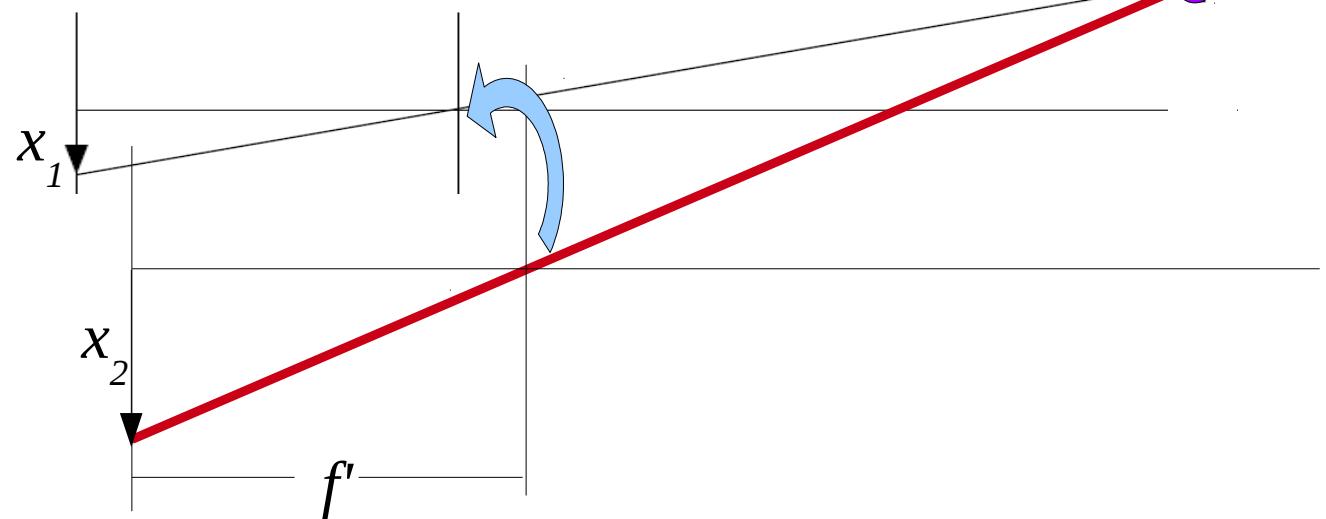
Motion Analysis

Projection of a scene point changes with:

1. object motion



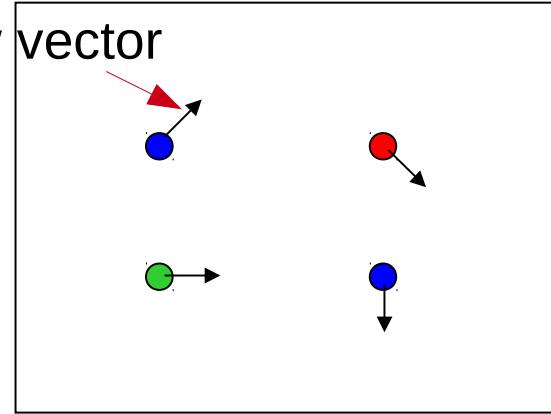
2. camera motion, or “ego motion”



Both types of movement give rise to “optic flow”.

Optic flow (OF)

optic flow vector



$I(x,y,t-1)$

$I(x,y,t)$

optic flow vector

the image motion of a scene point.

optic flow field

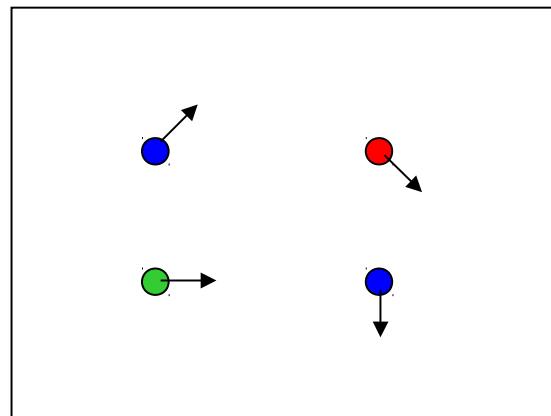
the collection of all the optic flow vectors

Optic flow fields can be sparse (vectors defined only for specified features) or dense (vectors defined everywhere).

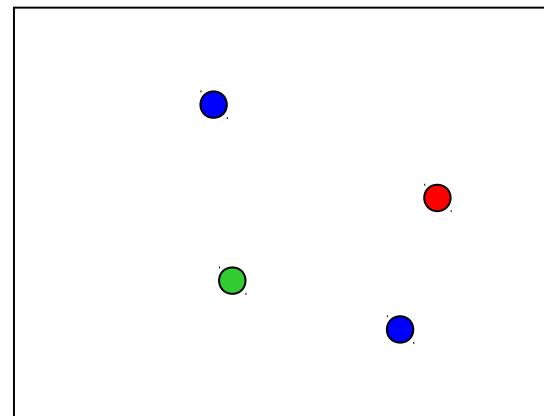
optic flow vectors are analogous to disparity vectors in stereo vision

measuring optic flow requires finding correspondences between images

Motion field (MF)



$I(x,y,t-1)$



$I(x,y,t)$

motion field

the true image motion of a scene point

i.e. the actual projection of the relative motion between the camera and the 3D scene

Optic flow provides an approximation to the motion field.

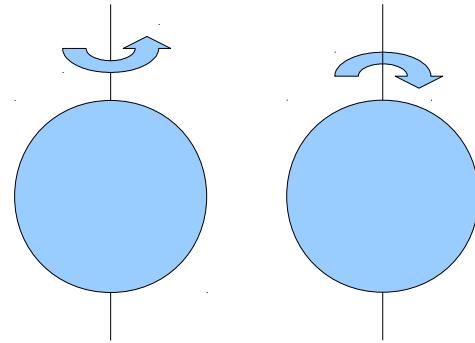
But it is not always accurate...

Motion field (MF) \neq Optic flow (OF)

Consider a smooth, **Lambertian**, uniform sphere rotating around a diameter:

MF \neq 0 as points on the sphere are moving

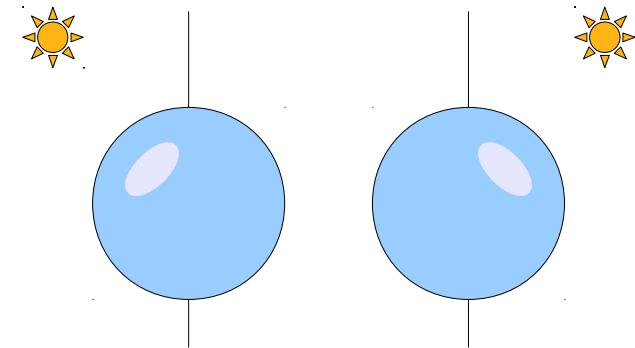
OF = 0 as there are no changes in the images



Consider a stationary, specular, sphere and a moving light source:

MF = 0 as points on the sphere are not moving

OF \neq 0 as there is a moving pattern in the images



Consider a barber's pole:

MF = horizontal

OF = vertical



Despite MF \neq OF in all circumstances, MF cannot be observed, so we must estimate MF by observing OF.

The video correspondence problem

To measure optic flow, it is necessary to find corresponding points in different frames of the video.

To solve the video correspondence problem, we can use:

- **Feature-based methods**

Extract descriptors from around interest points find similar features in next frame (identical to method used for stereo correspondence)

Sparse optic flow fields

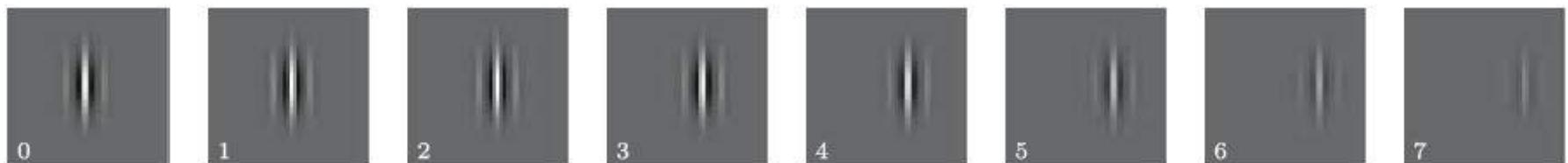
Suitable when image motion is large

- **Direct methods**

Directly recover image motion at each pixel from temporal variations of the image brightness (by applying **spatio-temporal filters**)

Dense optic flow fields, but sensitive to appearance variations

Suitable when image motion is small



The video correspondence problem

To measure optic flow, it is necessary to find corresponding points in different frames of the video.

To use feature-based methods

Basic requirements to be able to solve the correspondence problem:

1. Most scene points visible in both images
2. Corresponding image regions appear “similar”

Video Constraints on Correspondence

To measure optic flow, it is necessary to find corresponding points in different frames of the video.

Constraints used to help find corresponding points:

Spatial coherence

Similar neighbouring flow vectors are preferred over dissimilar ones.

- The assumption is that the scene is made up of smooth surfaces, and hence, neighbouring points in the scene typically belong to the same surface, and hence, typically have similar motions and induce similar optic flow.

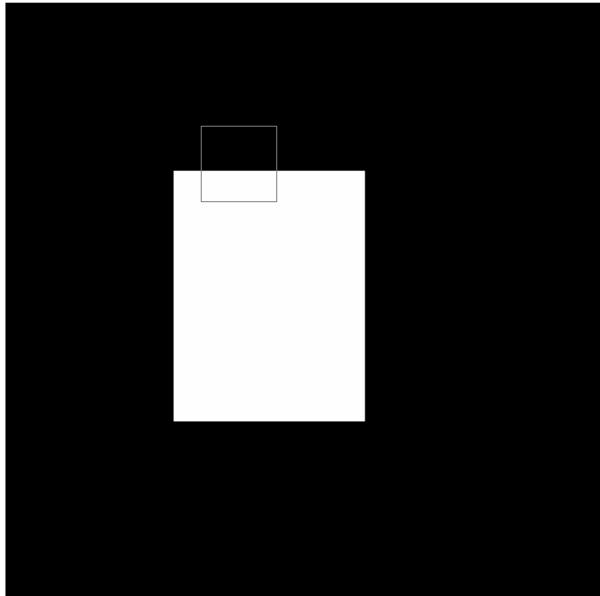
Small motion

Small optic flow vectors are preferred over large ones.

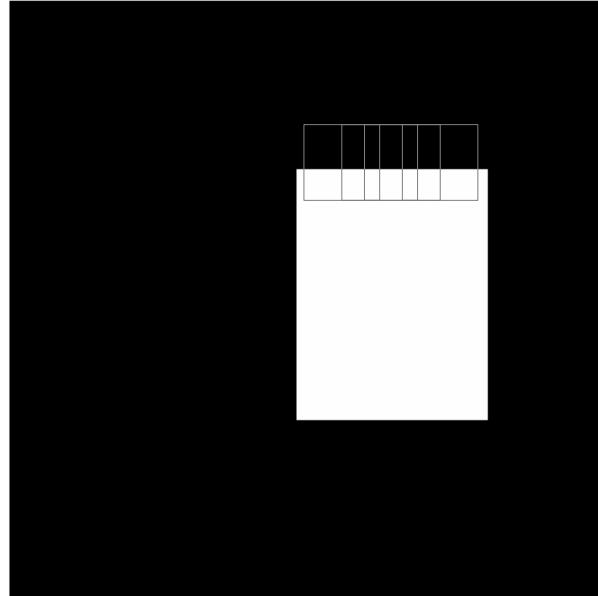
- The assumption is that relative velocities are slow compared to the frame rate, and hence, that the amount of motion between frames is small compared to the size of the image.

Aperture problem

Consider two consecutive frames showing a moving rectangle.



$I(x,y,t)$



$I(x,y,t+1)$

The image patch marked in the first frame could match any of the patches marked in the second frame.

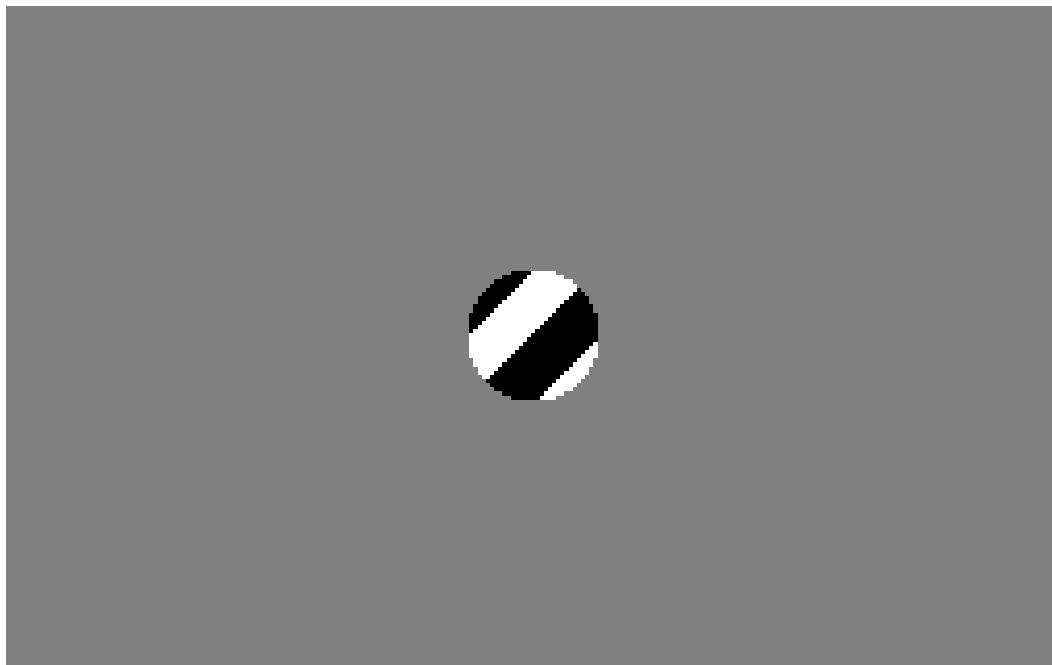
Because the intensity varies across the edge but not along the edge a motion parallel to the edge can never be recovered.

The inability to determine optic flow along the direction of the brightness pattern is known as the **“aperture problem”**.

Aperture problem

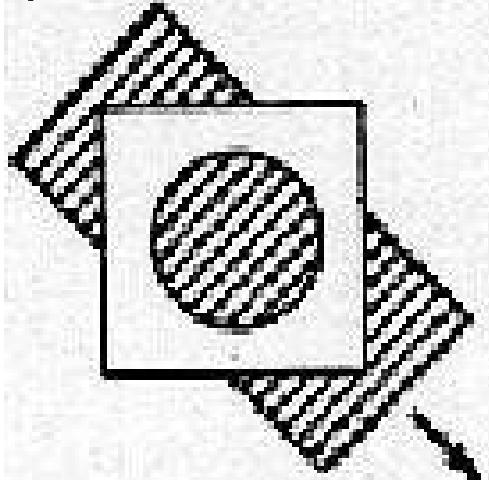
The brain is also faced with the aperture problem as each motion sensitive neuron sees only a small spatial region (i.e. its receptive field)

A demonstration.
What is the direction of motion here?

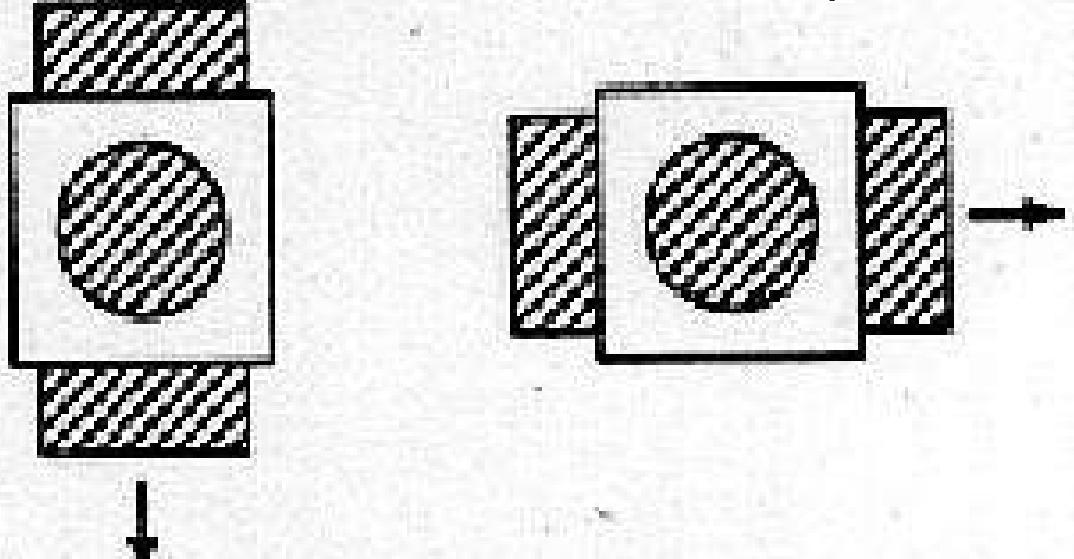


Aperture problem

This is what we perceive:



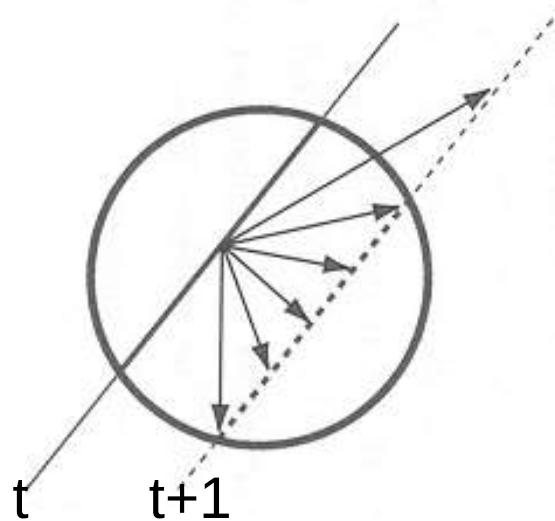
These motions are also possible:



Any movement with a component perpendicular to the edge is possible.

Locally the direction of motion is ambiguous.

We may see motion perpendicular to edge because:

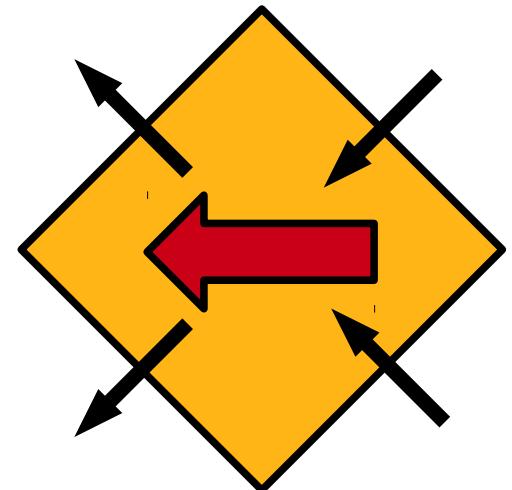
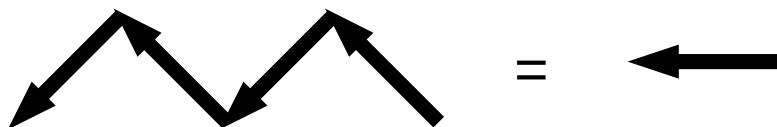


- it is average of all possibilities
- it predicts the slowest movement

Aperture problem: solutions

Local motion measurements are combined across space.

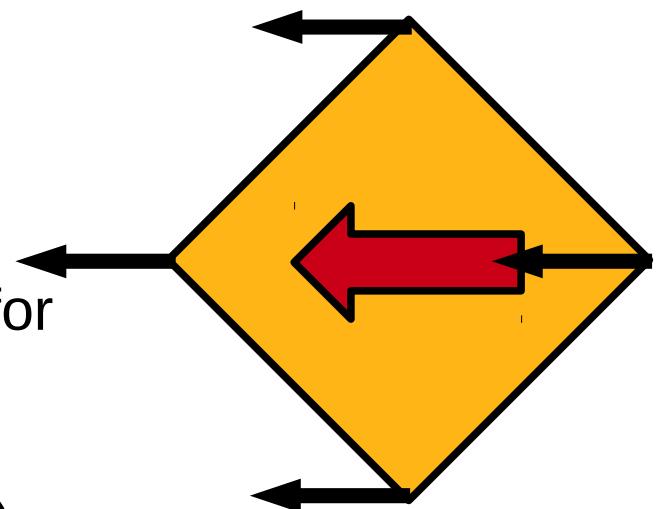
i.e. more than one local measurement is used to resolve the ambiguity and to accurately compute the direction of global motion.



Locations where the direction of motion is unambiguous are used.

i.e. corners

Note, SIFT and Harris corner detectors commonly used for finding interest points for solving stereo correspondence. The same features are also good for solving video correspondence (i.e. calculating optic flow)



Optic flow applications

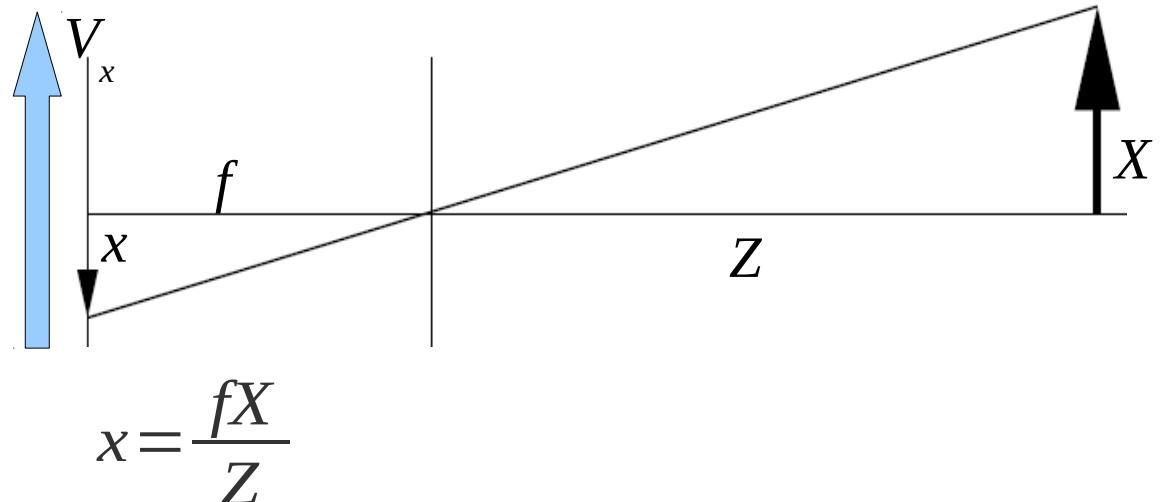
Optic flow can be used in various ways.

- To estimate the layout of the environment
 - depths and orientations of surfaces.
- To estimate ego motion
 - the camera velocity relative to a visual frame of reference.
- To estimate object motions
 - relative to the visual frame of reference, or relative to an environmental frame of reference.
- To obtain predictive information for the control of action. This information need not make layout or motion explicit.

Depth from optic flow and known ego-motion

Simple case 1:

- direction of motion is perpendicular to optical axis
- velocity V_x of camera is known



Given two images taken at times 1 and 2:

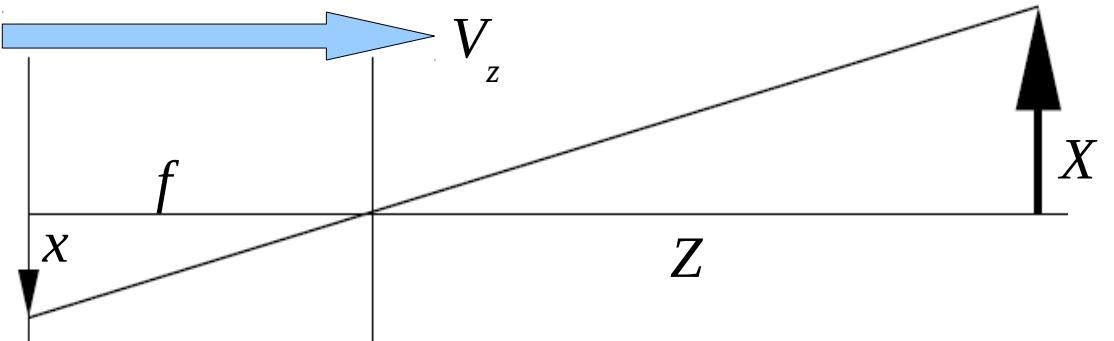
$$Z = \frac{fX_1}{x_1} = \frac{fX_2}{x_2} \quad \Rightarrow \quad X_1 x_2 = X_2 x_1$$
$$X_1 x_2 = (X_1 - V_x t) x_1$$
$$X_1 (x_2 - x_1) = -V_x t x_1$$
$$X_1 = \frac{-V_x x_1}{\dot{x}} \quad \text{where: } \dot{x} = \frac{(x_2 - x_1)}{t}$$
$$Z = \frac{fX_1}{x_1} = -\frac{fV_x}{\dot{x}}$$

Hence, by measuring the velocity of an image point, we can recover its depth.

Depth from optic flow and known ego-motion

Simple case 2:

- direction of motion is along camera optical axis
- velocity V_z of camera is known



$$x = \frac{fX}{Z}$$

Given two images taken at times 1 and 2:

$$fx = x_1 Z_1 = x_2 Z_2 \rightarrow x_1(Z_2 + V_z t) = x_2 Z_2$$

$$x_1 V_z t = (x_2 - x_1) Z_2 \rightarrow Z_2 = \frac{V_z x_1}{\dot{x}}$$

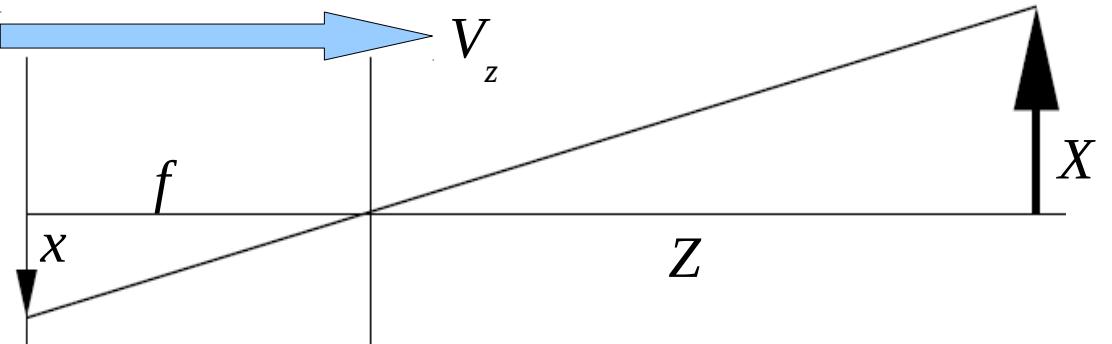
$$\text{where } \dot{x} = \frac{(x_2 - x_1)}{t}$$

Hence, by measuring the velocity of an image point, we can recover its depth.

Time-to-collision from optic flow

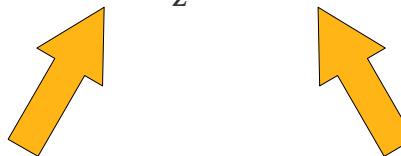
Simple case 2:

- direction of motion is along camera optical axis
- velocity V_z of camera is **unknown**



$$Z_2 = \frac{V_z x_1}{\dot{x}} \rightarrow \frac{Z_2}{V_z} = \frac{x_1}{\dot{x}}$$

= time-to-collision (if the camera velocity is constant).



can be measured purely from the image.

Time-to-collision from optic flow

$$\text{Time-to-collision} = \frac{x_1}{\dot{x}} = \frac{\alpha_1}{\dot{\alpha}} = \frac{2A_1}{\dot{A}}$$

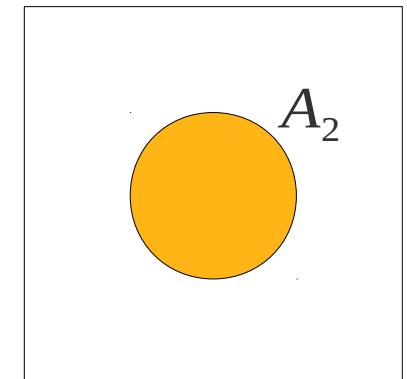
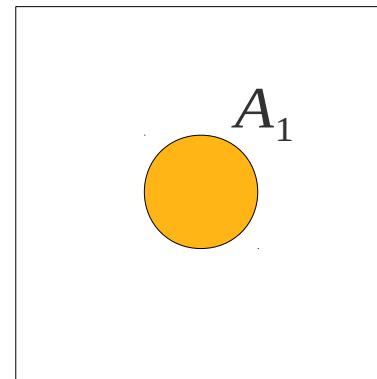
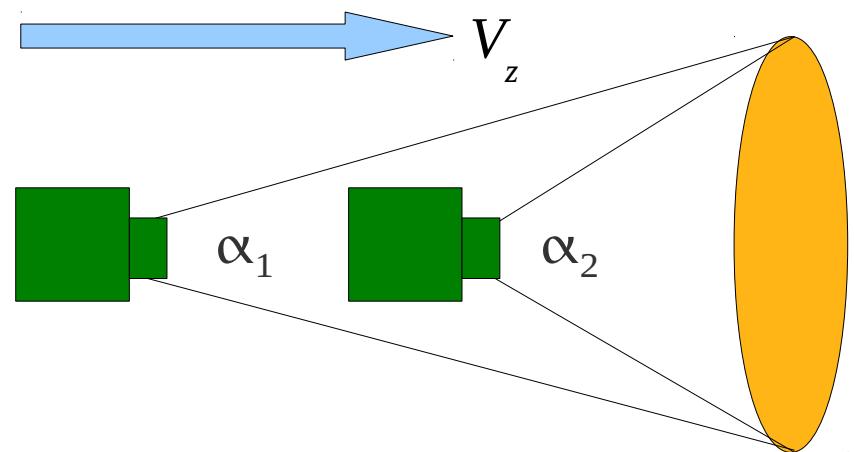
where:

α is the angle subtended by the object,
and

A is the area of the the object's image.

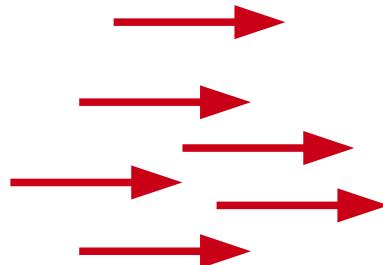
Hence, time-to-collision can be calculated without knowing anything about the speed of the camera, the size of, or distance from, the object.

Used in nature by birds and insects for catching prey, landing on a surface, etc.

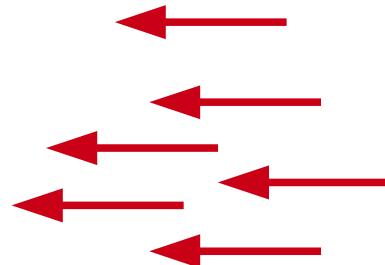


Ego-motion from optic flow

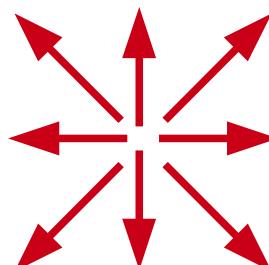
Camera translations induce characteristic patterns of optic flow (for a static scene).



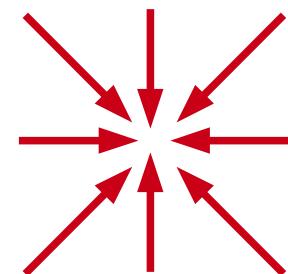
turn/translate
left



turn/translate
right



move
forward



move
backward

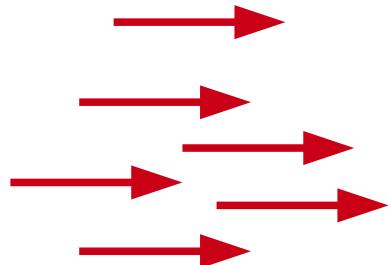
Parallel Optic Flow Field ($V_z = 0$)

- all optic flow vectors are parallel
- direction of camera movement opposite to direction of optic flow field
- speed of camera movement proportional to length of optic flow vectors

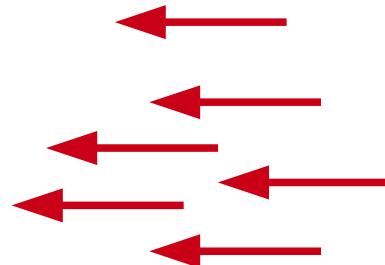
Radial Optic Flow Field ($V_z \neq 0$)

- all optic flow vectors point towards/away from a vanishing point (p_0)
- direction of camera movement determined by whether FOE (focus of expansion) or FOC (focus of contraction)
- destination of movement is **FOE**

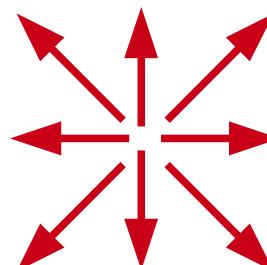
Relative depth from optic flow



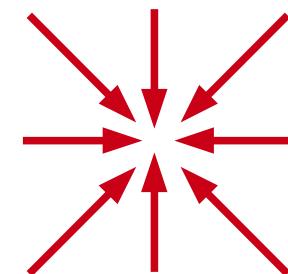
turn/translate
left



turn/translate
right



move
forward



move
backward

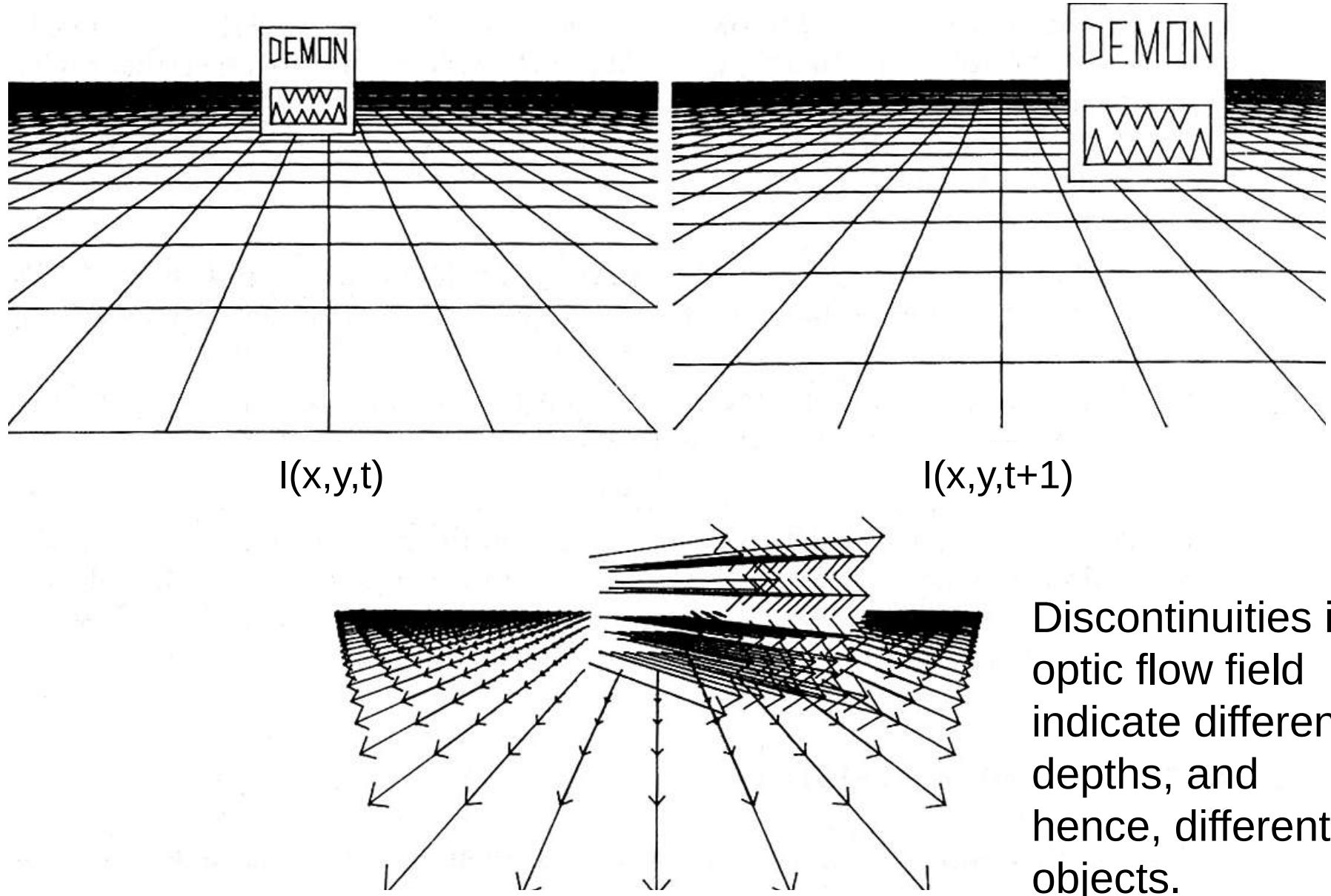
Parallel Optic Flow Field ($V_z = 0$)

- depth inversely proportional to magnitude of optic flow vector
- This is the same as motion **parallax** with fixation on infinity

Radial Optic Flow Field ($V_z \neq 0$)

- depth of point p inversely proportional to magnitude of optic flow vector, and also proportional to distance from p to p_0

Segmentation from optic flow



Optic flow applications

Using optic flow you can

- get to a destination by moving so that the destination is the FOE
- judge relative depths by relative magnitudes of optic flow vectors (points closer to the camera move more quickly across the image plane)
- measure absolute depths (with knowledge of camera velocity)
- judge camera speed by the rates of expansion/contraction
- measure time-to-collision
- judge direction of ego-motion
- judge directions and speeds of object motions
- segment objects at different depths / determine orientation of surfaces

Tracking

When **high-level features**, such as objects, are matched across several frames, the algorithms are usually referred to as tracking algorithms rather than optic flow algorithms.



Tracking

When **high-level features**, such as objects, are matched across several frames, the algorithms are usually referred to as tracking algorithms rather than optic flow algorithms.

Tracking algorithms use previous frames to **PREDICT** location of object in next frame



Intent:

- Do less work looking for the object, restrict the search.
- Get improved estimates since measurement noise is averaged out.

Methods:

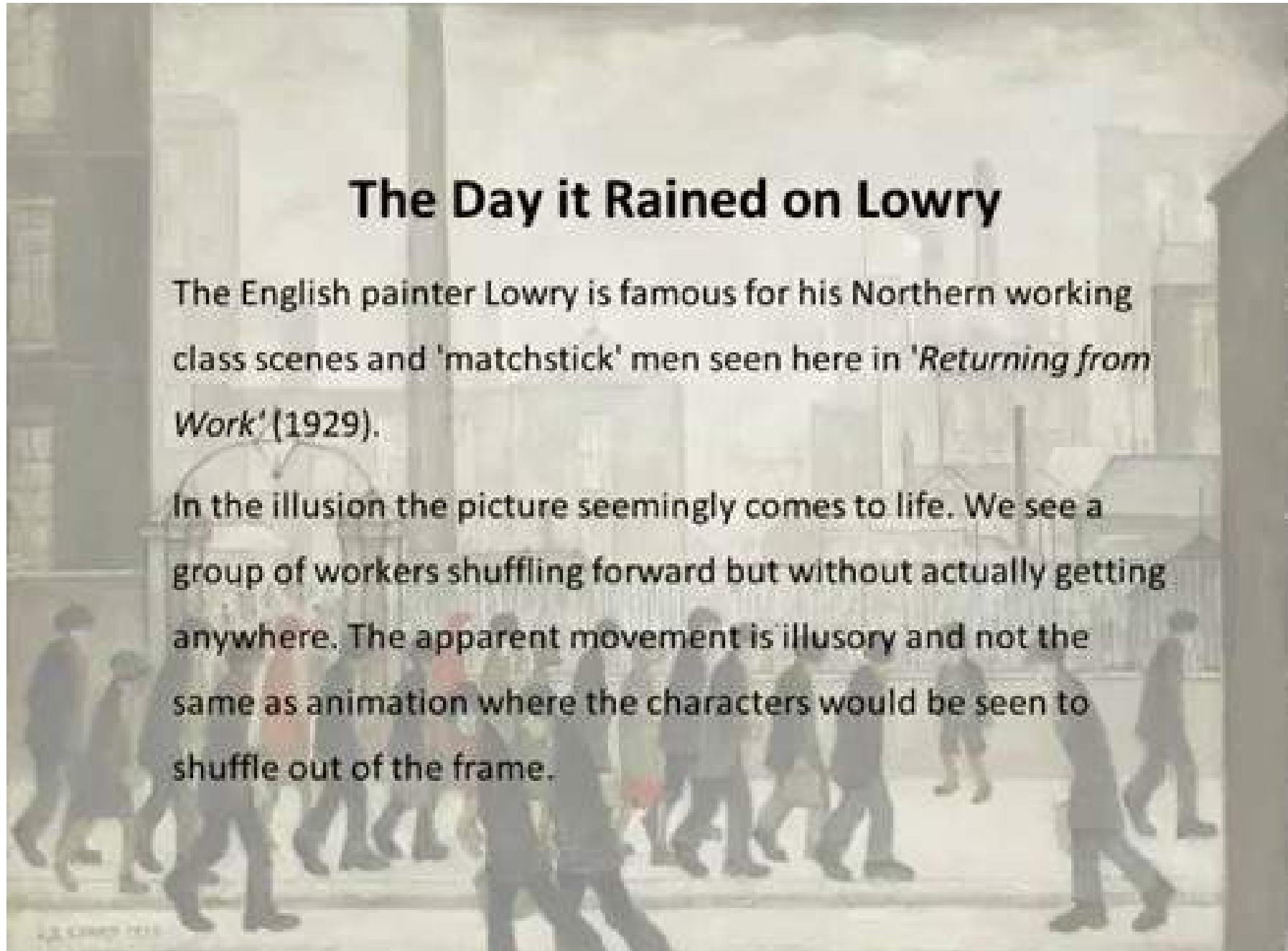
- Kalman filtering
- Particle filtering

Tracking

Humans **PREDICT** movement

- Extrapolating object trajectory from previous locations
- But also, from high-level knowledge of typical movement

Tracking



The Day it Rained on Lowry

The English painter Lowry is famous for his Northern working class scenes and 'matchstick' men seen here in '*Returning from Work*' (1929).

In the illusion the picture seemingly comes to life. We see a group of workers shuffling forward but without actually getting anywhere. The apparent movement is illusory and not the same as animation where the characters would be seen to shuffle out of the frame.

Segmentation from motion

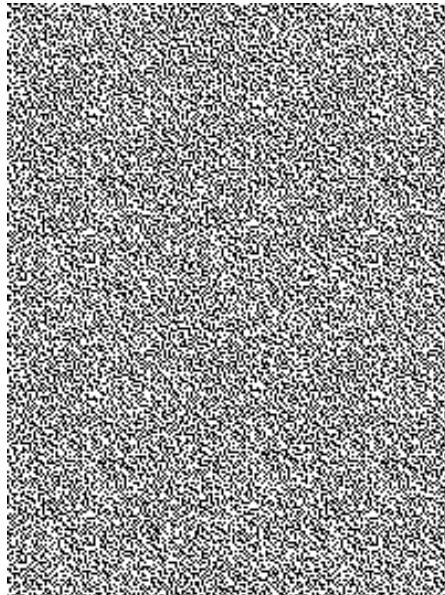
Motion → optic flow discontinuities → segmentation

Motion → optic flow → depth → segmentation

Motion → segmentation

e.g. camouflaged objects are more easily seen when they move

i.e. Gestalt law of common fate



Segmentation: image differencing

For a static camera, image differencing can be used to segment moving objects from static ones.

Successive images are subtracted pixel by pixel, and a binary image containing absolute differences exceeding some threshold is generated.

The intensity level changes most in regions where there is motion, and hence these regions are detected.

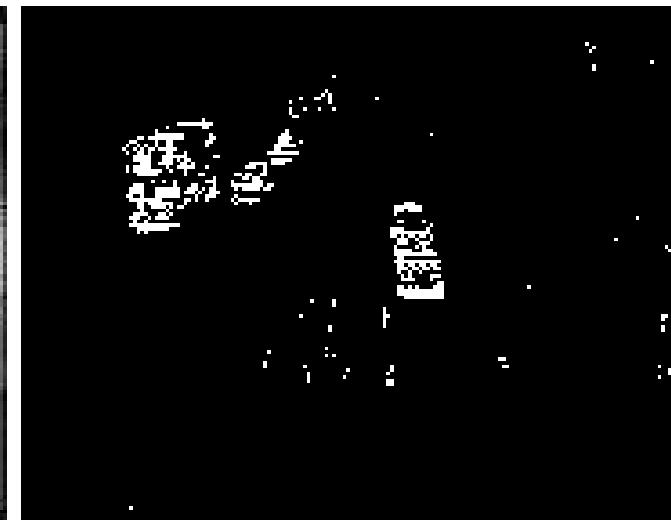
$I(x,y,t)$



$I(x,y,t+1)$



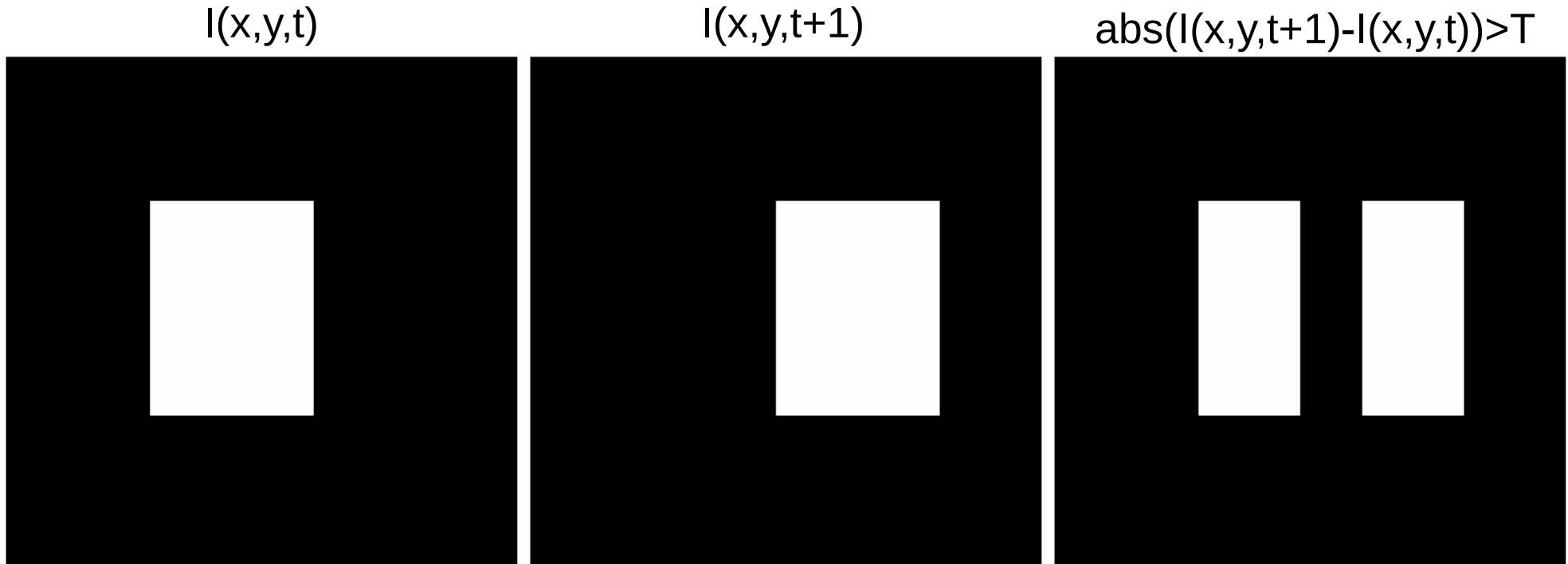
$\text{abs}(I(x,y,t+1)-I(x,y,t))>T$



Segmentation: image differencing

One problem with this technique is that the grey level changes most where background is replaced by the object (or vice versa), and may not change in locations occupied by the object in both frames.

Thus there is a kind of double view of each object.



Segmentation: background subtraction

To avoid this problem learn the static part of the scene and use this reference image as the background to be subtracted.

Now objects that are temporarily stationary are still seen as foreground (e.g. car at bottom-left of example).

Still missing pixels and noise – **morphological operations** (e.g. dilation and erosion – see lecture on image segmentation) can be used to clean up result.

$B(x,y)$



$I(x,y,t)$



$\text{abs}(I(x,y,t)-B(x,y))>T$



Segmentation: background subtraction

How to learn the static part of the scene?

Adjacent Frame Difference

$$B(x, y) = I(x, y, t-1)$$

- Each image is subtracted from previous image in sequence (i.e. image differencing)

Off-line average

$$B(x, y) = \frac{1}{N} \sum_{t=1}^N I(x, y, t)$$

- Pixel-wise mean values are computed during separate training phase (also called Mean and Threshold)

Moving average

$$B(x, y) \leftarrow (1 - \beta) B(x, y) + \beta I(x, y, t)$$

- Background model is linear weighted sum of previous frames. Can overcome problems of slow illumination changes, so most typically used.

Segmentation: background subtraction

Hence, a typical algorithm for motion segmentation is:

for each frame ($t = 1:N$)

 Update background model

$$B(x, y) \leftarrow (1 - \beta) B(x, y) + \beta I(x, y, t)$$

 Compute frame difference

$$\delta(x, y, t) = \|I(x, y, t) - B(x, y)\|$$

 Threshold frame difference

$$\delta_T(x, y, t) = \delta(x, y, t) > \text{thres}$$

 Noise removal

$$\delta_T(x, y, t) = \text{close}(\delta_T(x, y, t))$$

end

Objects are detected where $\delta_T(x, y, t)$ is non-zero

Segmentation: examples

	Time of Day	Light Switch	Waving Trees	Camouflage	Bootstrapping	Foreground Aperture
						
Light gradually brightened						
Ideal						
Image differencing						
Off-line average						
Moving average						

Segmentation: background subtraction

How to learn the static part of the scene?

Ideally background image should:

1. exclude “interesting” changes so that these are not subtracted from the image:
 - e.g. objects (even if these stop temporarily)
2. incorporate “uninteresting changes” so that these are subtracted from image:
 - illumination changes (e.g. due to sun/clouds, lights being switched on/off, shadows)
 - static objects that deform (e.g. trees blowing in wind)

Summary: Optic flow

Aim: to infer properties of the 3D scene from 2 or more images taken at different times. Two sub-problems:

1. Correspondence. Determining which points in the different frames are projections of the same point in the scene.

Hence, calculate optic flow (e.g. motion disparities).

Constraints:

- Small motion (extent of search reduced by expectation that motion between frames is small)
- Spatial coherence (optic flow varies smoothly assuming continuous surfaces)

Summary: Optic flow

2. Reconstruction. Given the correspondence between points, calculate 3D structure and motion of the observed scene.

- with knowledge of ego-motion: calculate absolute depth
- without knowledge of ego-motion:
 - calculate relative depths
 - time-to-collision
 - direction of ego-motion
 - heading of ego-motion

Summary: Segmentation

Segment moving objects from stationary background.

Calculate: $abs(I(x,y,t)-B(x,y))>T$

– *image differencing*

$$B(x, y) = I(x, y, t-1)$$

– *background subtraction*

» *off-line average*

$$B(x, y) = \frac{1}{N} \sum_{t=1}^N I(x, y, t)$$

» *moving average*

$$B(x, y) \leftarrow (1-\beta) B(x, y) + \beta I(x, y, t)$$

Computer Vision (7CCSMCVI / 6CCS3COV)

Recap

- **Image formation**
- **Low-level vision**
- **Mid-level vision**
 - Segmentation and grouping
 - Correspondence problem
 - Stereo and Depth
 - Video and Motion
- **High-level vision**
 - Object recognition

← Today

Today

- What is Object Recognition
 - Identification
 - Categorisation
 - Localisation
- Methods for performing Object Recognition
 - template matching
 - sliding window
 - edge matching
 - model-based
 - intensity histograms
 - implicit shape model
 - SIFT feature matching
 - bag-of-words
 - geometric invariants

Object recognition tasks

Identification

Determine identity of an individual instance of an object



vs



Clinton

Bush



vs



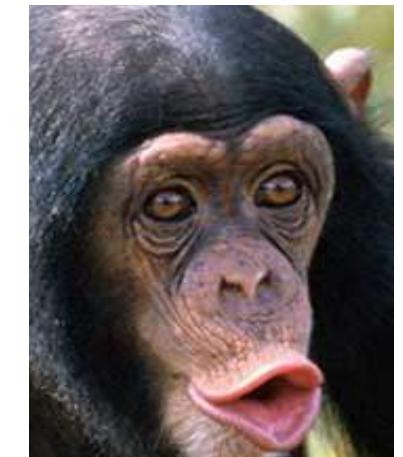
Samsung Galaxy On8

iPhone 7 Plus

Object recognition tasks

Classification

Determine the category of an object



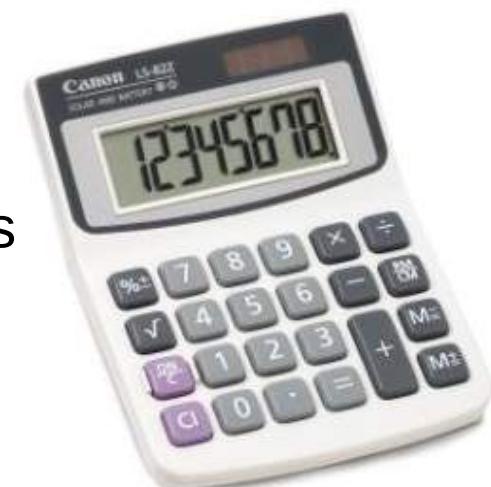
vs

Human

Chimpanzee



vs



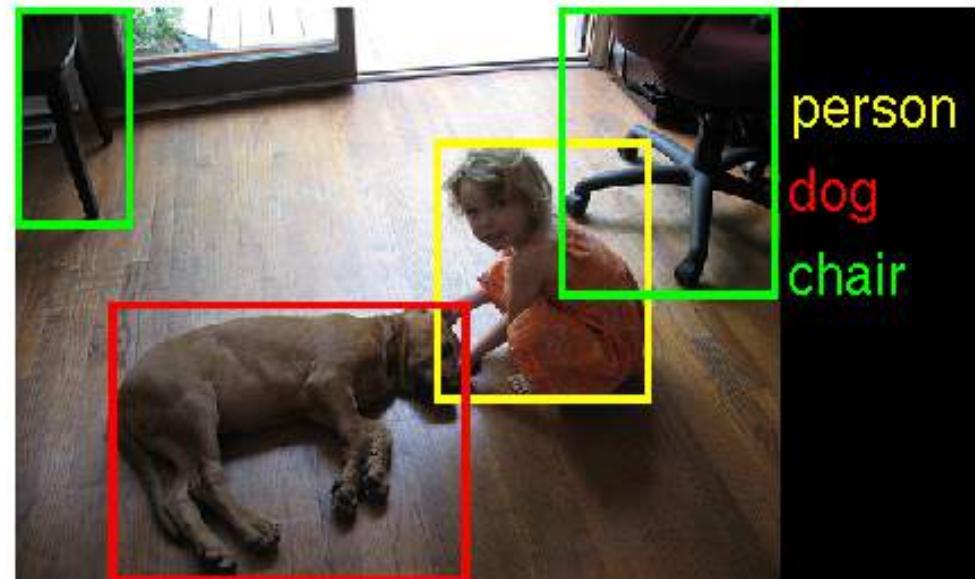
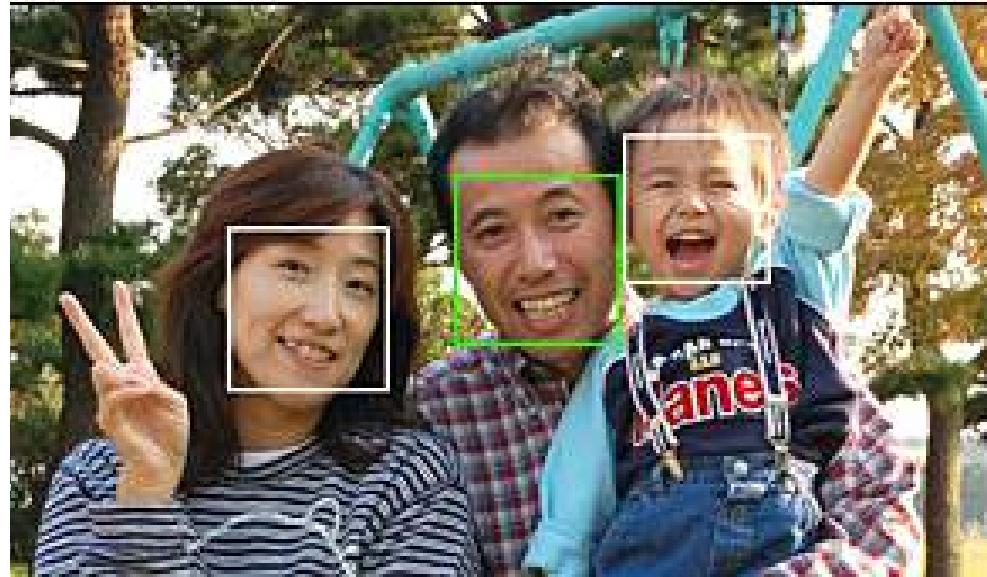
Telephone

Calculator

Object recognition tasks

Localisation

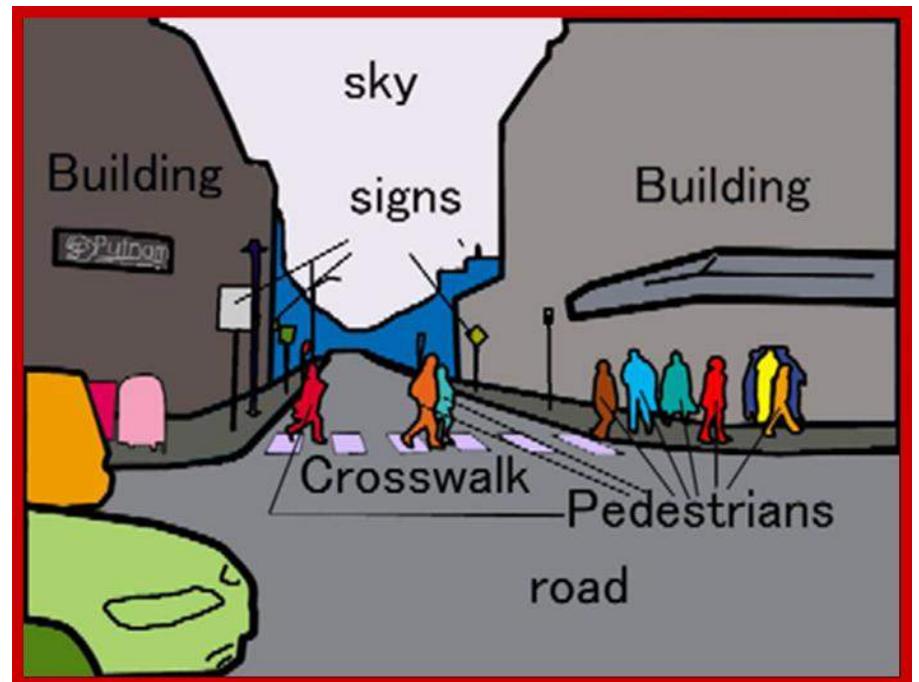
Determine presence of and/or location of object in an image



Object recognition tasks

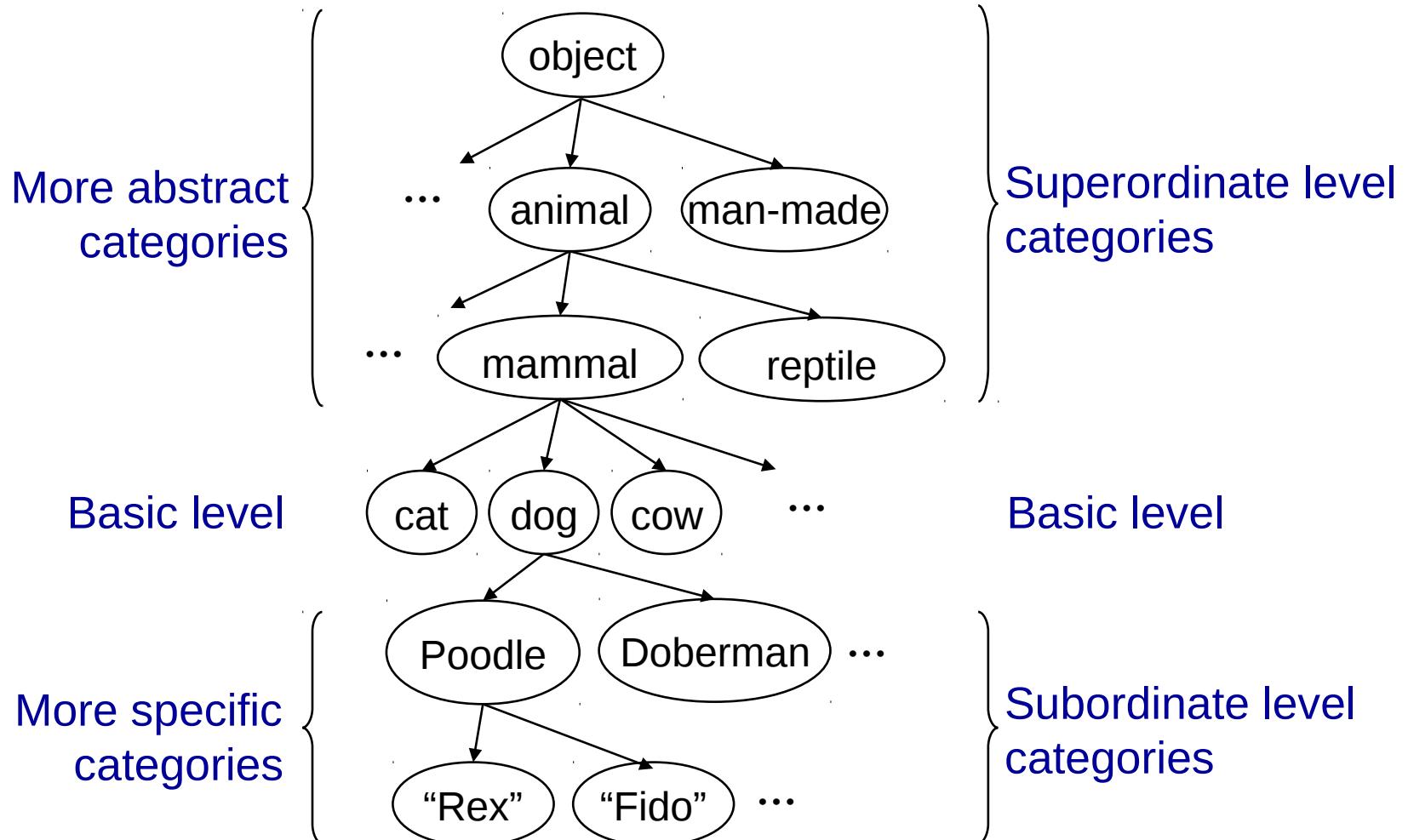
Localisation

If localisation is sufficiently fine grained and for a sufficiently large number of categories, then results are like image segmentation (called “semantic segmentation”)



Category hierarchy

Classification can take place across a hierarchy of different category types.



Identification is classification at one extreme in this hierarchy.

Category hierarchy

The basic level has a special significance for human object recognition.

It is the level that:

- humans are usually fastest at recognizing category members.
- humans usually start with basic-level categorization *before* doing identification.
- is first named and understood by children.

Category hierarchy

The basic level is the:

- highest level at which category members share many common features
 - i.e. basic level is perceptually homogeneous
 - e.g. apples are similar shape, size, texture, colour, etc.
 - c.f. the next level up, fruits, don't have many features in common
- lowest level at which category members have features distinct from other categories at the same level
 - i.e. basic level is relatively easy to discriminate
 - e.g. apple vs banana vs grape, etc.
 - c.f. the next level down, Bramley vs Coxs' vs Granny Smiths

Object recognition requirements

Object recognition requires:

Sensitivity to (possibly small) image differences relevant to distinguishing one object identity/category from another.



Insensitivity or tolerance to (possibly large) image differences that do not affect object identity or category.



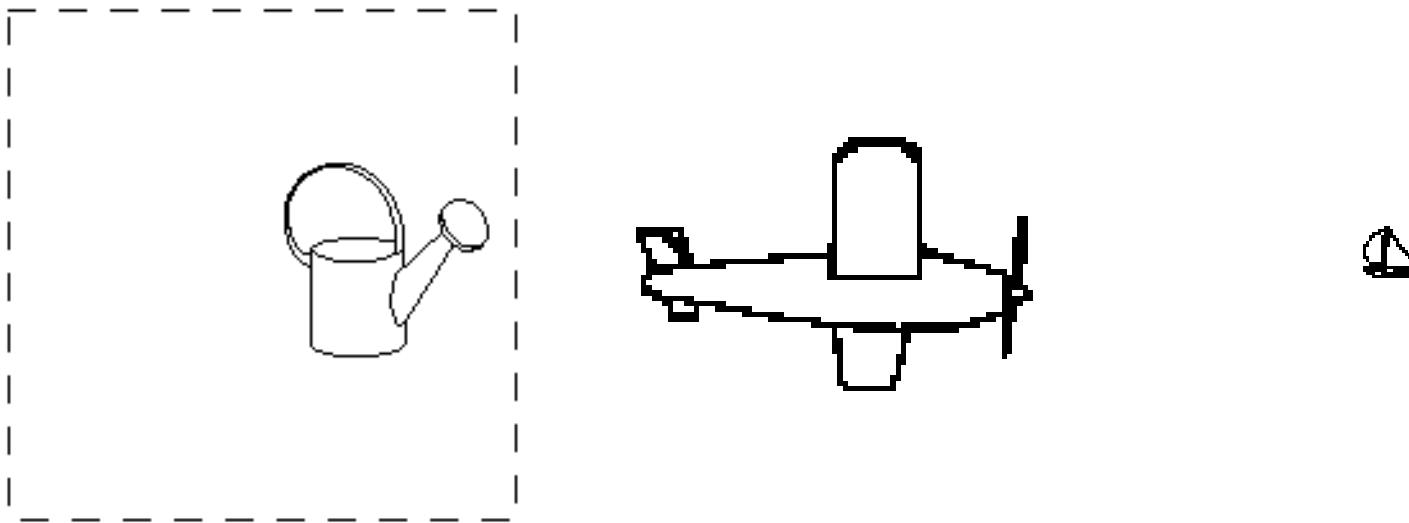
Object recognition requirements

e.g. Insensitivity to “background” clutter and occlusion.



Object recognition requirements

e.g. Insensitivity to viewpoint



e.g. Insensitivity to lighting



Object recognition requirements

e.g. Insensitivity to non-rigid deformations



e.g. Insensitivity to within category variation



Object recognition procedure

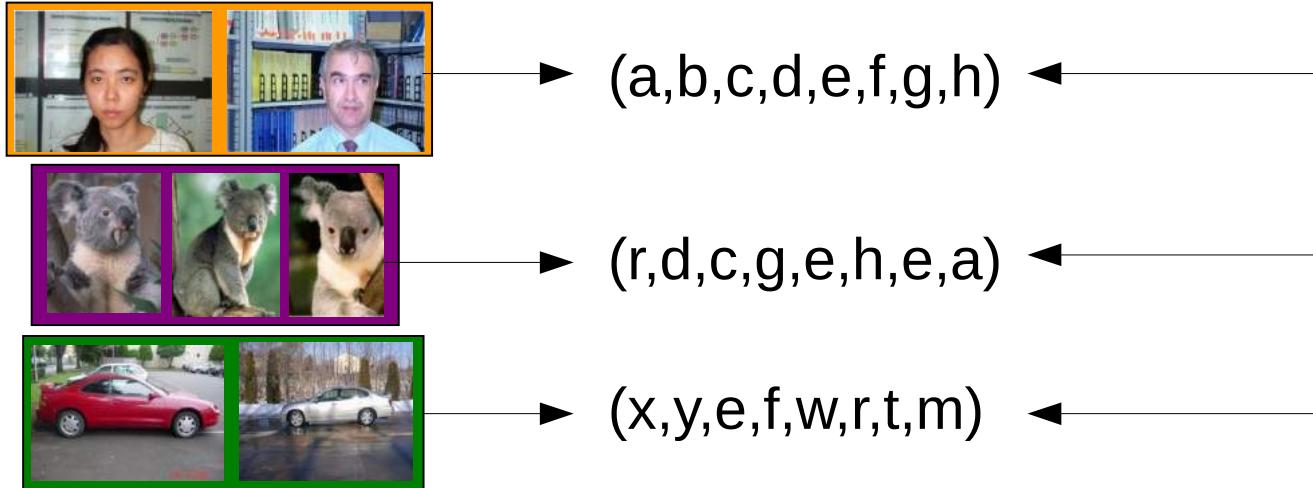
Associating information extracted from images with objects

- Requires image data
- Requires *representations* of objects
- Requires *matching* techniques

Object recognition procedure

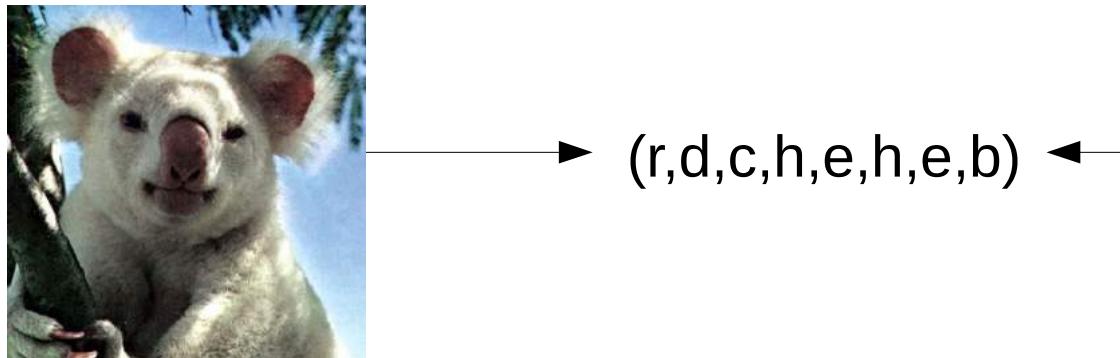
Off-line:

Extract representations from training examples



On-line:

Extract representation from input image



Match image
with training
examples to
determine
object class
or identity

Object recognition procedure

Methods vary
in terms of:

representation used:

local vs global features;
2D vs 3D; pixel
intensities vs other
features



(a,b,c,d,e,f,g,h)



(r,d,c,g,e,h,e,a)



(x,y,e,f,w,r,t,m)



(r,d,c,h,e,h,e,b)

matching
procedure:
top-down vs
bottom-up;
measure of
similarity,
classification
criteria

Template matching

Representation:

A template is an image of the object that is to be recognized (i.e. an array of pixel intensities).

Matching:

For every template:

- Search every image region
- Calculate similarity between template and image region

Choose the “best” match, or
all matches where similarity
exceeds a threshold



template:

$$\text{smile} = I_1$$

image region:

$$\text{smile} = I_2$$

Similarity Measures

We can **maximise** the following measures:

Cross-correlation:

$$\sum_{i,j} I_1(i,j)I_2(i,j)$$

If I_1 and I_2 are considered to be vectors in feature space, then the cross-correlation is the dot-product of these vectors $I_1 \cdot I_2 = \|I_1\| \|I_2\| \cos \theta$

Note: template matching using cross-correlation can be implemented using convolution and a rotated template.

Normalised cross-correlation: (NCC) $\frac{\sum_{i,j} I_1(i,j)I_2(i,j)}{\sqrt{\sum_{i,j} I_1(i,j)^2} \sqrt{\sum_{i,j} I_2(i,j)^2}} = \frac{I_1 \cdot I_2}{\|I_1\| \|I_2\|} = \cos \theta$

Correlation coefficient: $\frac{\sum_{i,j} (I_1(i,j) - \bar{I}_1)(I_2(i,j) - \bar{I}_2)}{\sqrt{\sum_{i,j} (I_1(i,j) - \bar{I}_1)^2} \sqrt{\sum_{i,j} (I_2(i,j) - \bar{I}_2)^2}}$

equals normalised cross-correlation if means are zero

Similarity Measures

We can **minimise** the following measures:

Sum of Squared Differences (SSD): $\sum_{i,j} (I_1(i,j) - I_2(i,j))^2$

Euclidean distance: $\sqrt{SSD} = \sqrt{\sum_{i,j} (I_1(i,j) - I_2(i,j))^2}$

Sum of Absolute Differences (SAD): $\sum_{i,j} |I_1(i,j) - I_2(i,j)|$

Template matching

To recognise multiple objects use multiple templates.



mouth template



eye template



nose template

Template matching: example

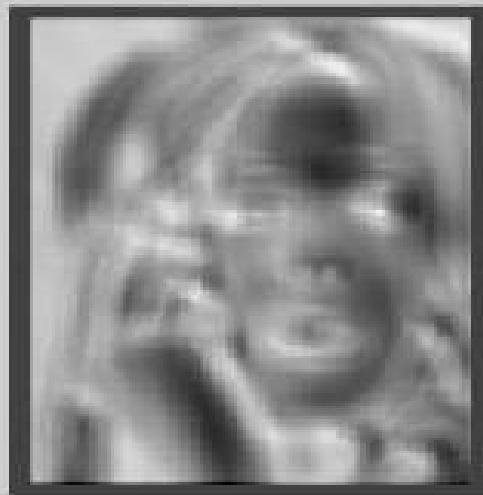
Template of right eye is flipped and used to locate left eye



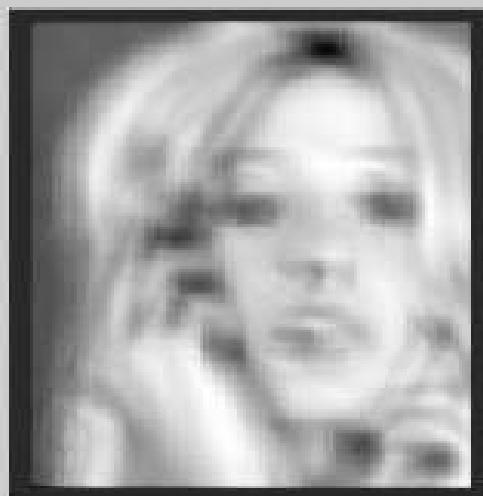
original image



darkened image



Sum of absolute
differences



note SAD peak over
the left eye (cross in
left image)

note SAD peak over
forehead (cross in left
image)

Template matching: example

Template of right eye is flipped and used to locate left eye



original image



darkened image



Normalised
cross-correlation



note NCC peak over
the left eye (cross in
left image)

note NCC peak over
the left eye (cross in
left image)

Template matching: problems

Distinguishing true matches from false matches



What constitutes a match?
How many peaks?

Template matching: problems

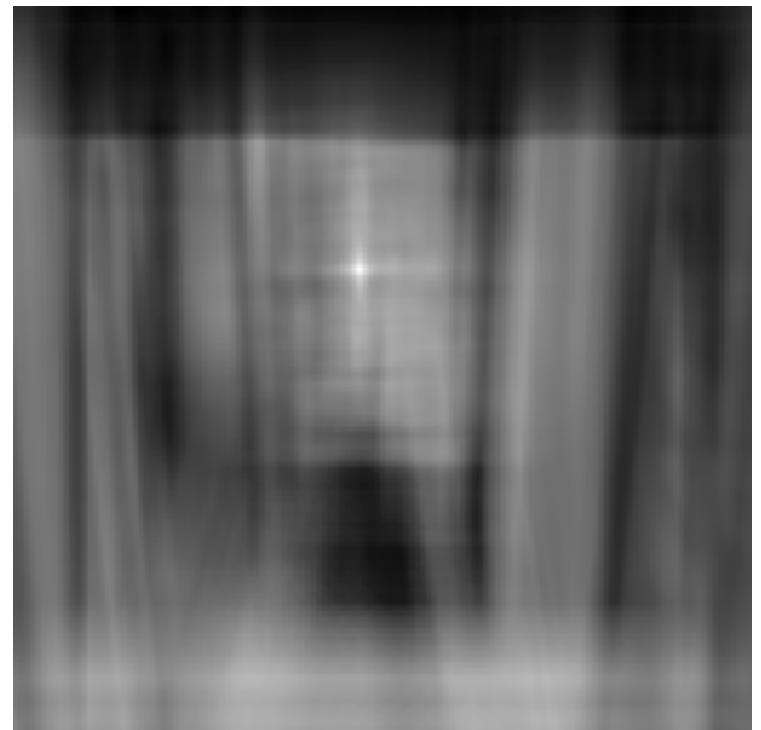
Template



Image



NCC



Template matching: problems

Template



Image



NCC



Template needs to be very similar to the target object

If an object appears scaled or rotated in the image, the match with its template will not be good.

Template matching: problems

Hence, to provide tolerance to viewpoint / within category variation it is necessary to use multiple templates for each object.



Template matching: problems

With so many comparisons, the threshold needs to be **high**, otherwise we will almost always find **false matches**.

However, because any deviation between the template and the image patch will result in a weak match, the threshold needs to be **low** to find all the **true matches**.



Template matching: problems

Tractability is an issue, especially if we need to deal with a wide range of variations in appearance.

e.g. for a small 250 by 200 pixel image, if we need to recognize 5 objects, each of which can occur at 30 viewpoints and 5 scales then we need to perform 37.5 million matches!



Template matching: problems

If an object appears occluded, then this may result in a template failing to match.
i.e. template matching is sensitive to occlusion.



Template matching: problems



The underlying issue is that the metric used for comparison is fundamentally not robust to changes in appearance between the template and the image patch.

- even small changes in scale, orientation, and viewpoint, as well as changes due to within class variation, non-rigid deformations, or occlusion will result in weak matches (indistinguishable from false matches)

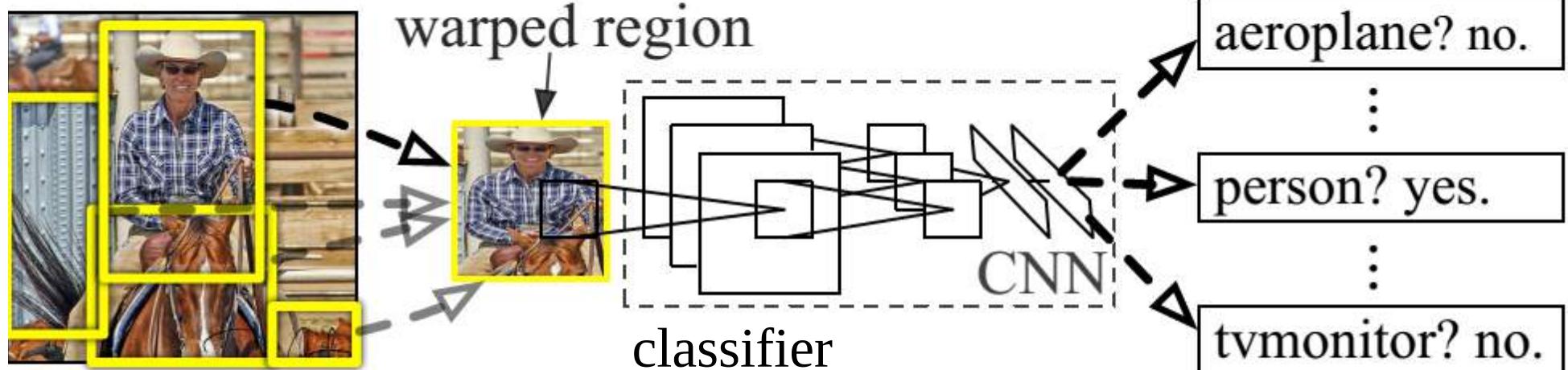
Hence

- invariance is difficult to achieve

Sliding window

Like template matching, except:

- 1) For each image patch, use a **classifier** to determine if it contains the object (i.e. replace simple comparison of intensity values with method that is more tolerant to changes in appearance).
- 2) Choose image patches of different shapes and sizes and resize them before inputting to the classifier (to also increase tolerance to changes in appearance).

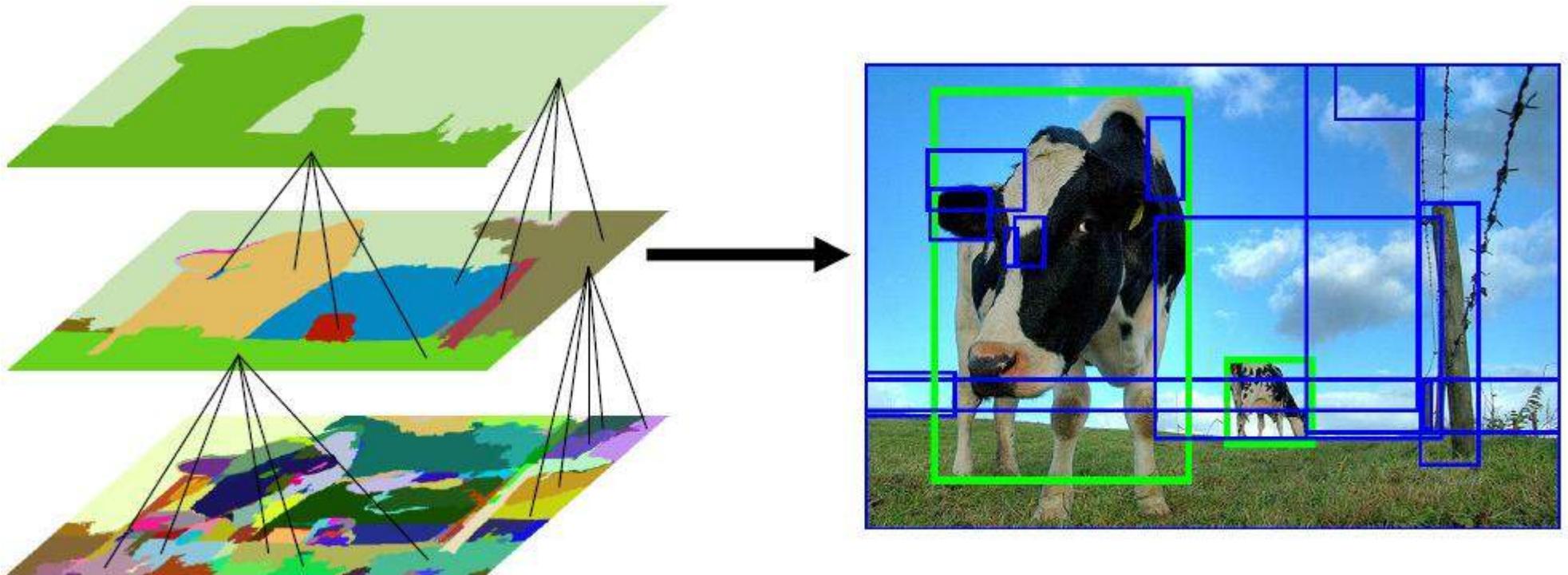


Sliding window

Classifying every image region of every possible size and shape (typically 100k+ regions) is very computationally expensive!

To improve tractability, images can be pre-processed to select regions (typically 1k+ regions) that are good candidates to contain an object.

This pre-processing is typically done using image segmentation.



Edge matching

Representation:

Like template matching, but template and input images pre-processed to extract edges



Input image

Matching:

For every edge template:

Search every image region

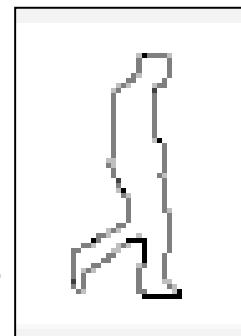
Calculate average of the minimum distances between points on the edge template (T) and points on the edge image (I)

$$D(T, I) = \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

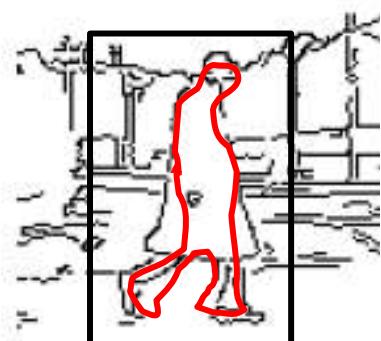


Edges detected

Choose the minimum value as best match



Template shape



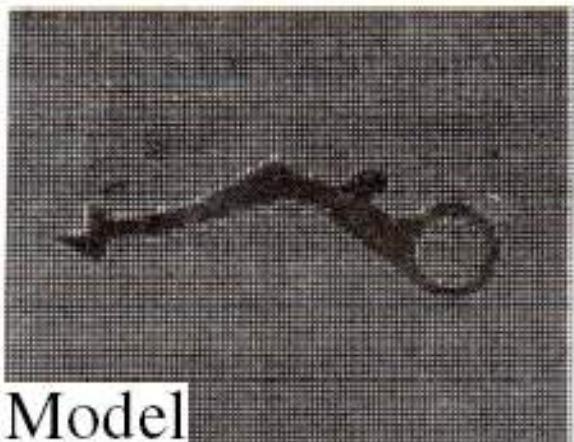
Best match

Model-based object recognition

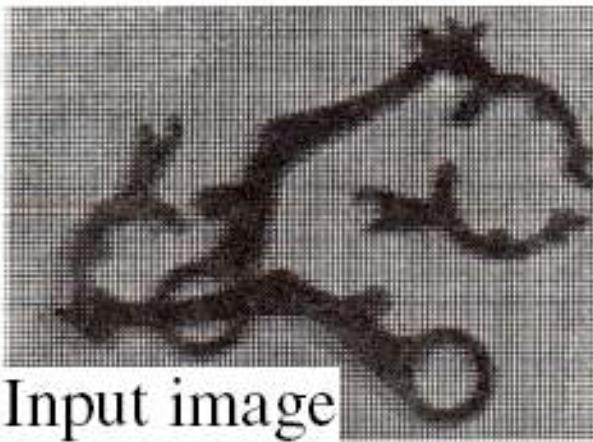
General idea

- Hypothesize object identity and pose
- Render object in image (“back-project”)
- Compare to image

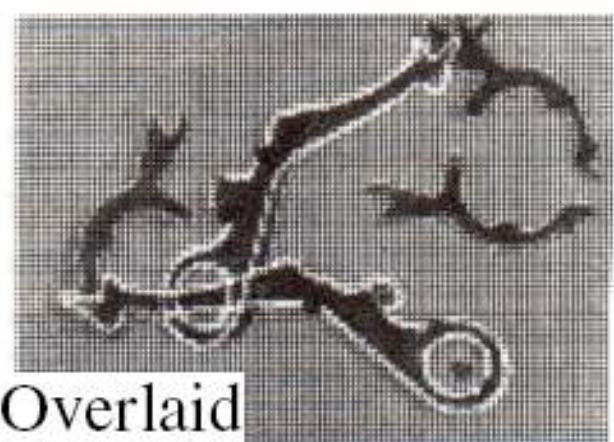
e.g. 2D:



Model



Input image



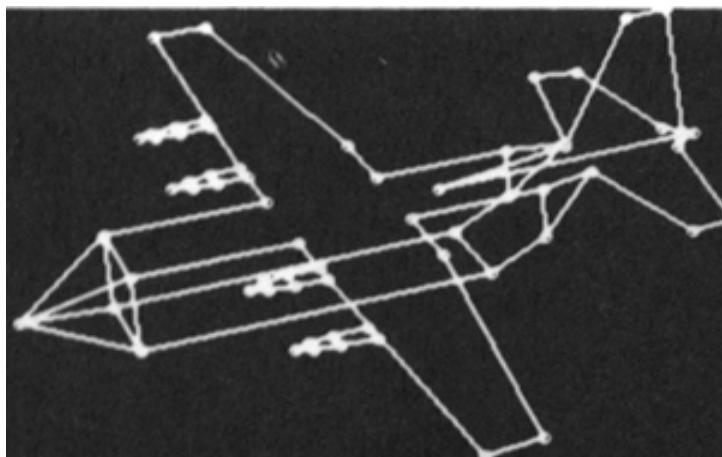
Overlaid

Model-based object recognition

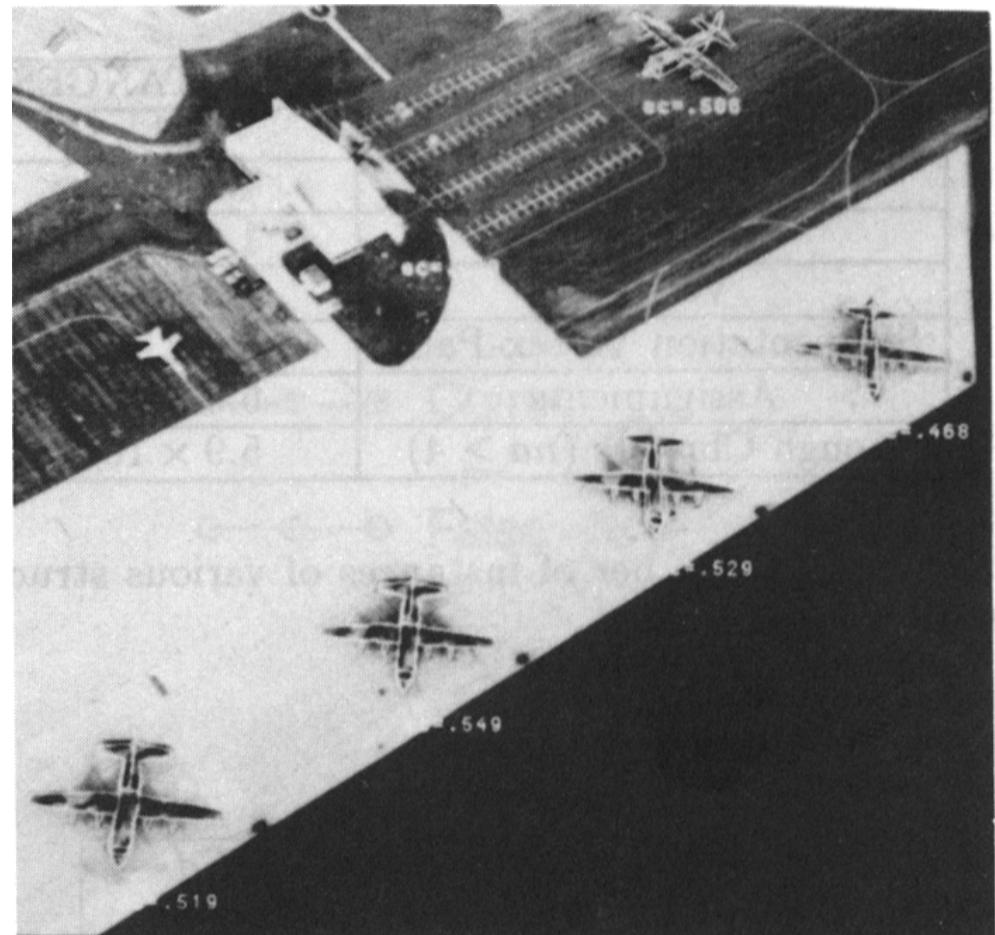
General idea

- Hypothesize object identity and pose
- Render object in image (“back-project”)
- Compare to image

e.g. 3D:



model



Model-based object recognition

Representation:

2D or 3D model of object shape

Matching:

Comparison of **back-projected** model with image, using:

- Edge score
 - are there image edges near predicted object edges?
 - very unreliable; in texture, answer is usually yes
- Oriented edge score
 - are there image edges near predicted object edges with the right orientation?
 - better, but still hard to do well

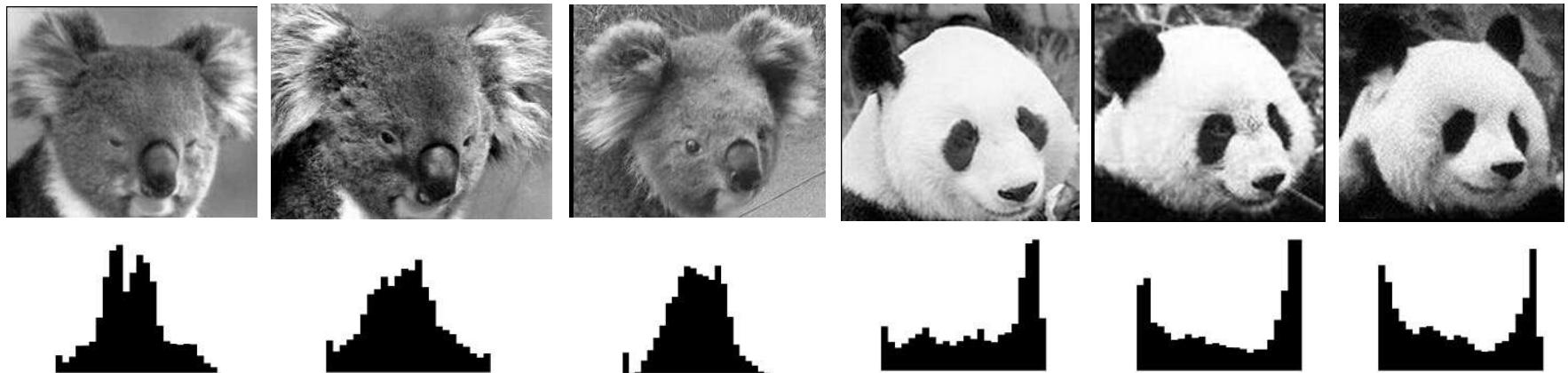
Intensity histograms

Representation:

Histogram of pixel intensity values (either greyscale or colour).

Matching:

Compare histograms to find closest match



- ✓ Histogram is fast and easy to compute.
- ✓ Insensitive to small viewpoint changes (unlike templates)
 - ✗ Sensitive to illumination and intra-class appearance variation
 - ✗ Insensitive to different spatial configurations (as spatial information not represented)

Intensity histograms

All these images have the same colour histograms



Insensitivity to viewpoint
changes (useful)

Insensitivity to spatial
configuration (not
always useful)

Intensity histograms

Skin has a very small range of (intensity independent) colours.
Hence, colour histograms are often used as part of face detection/recognition algorithms.



original images

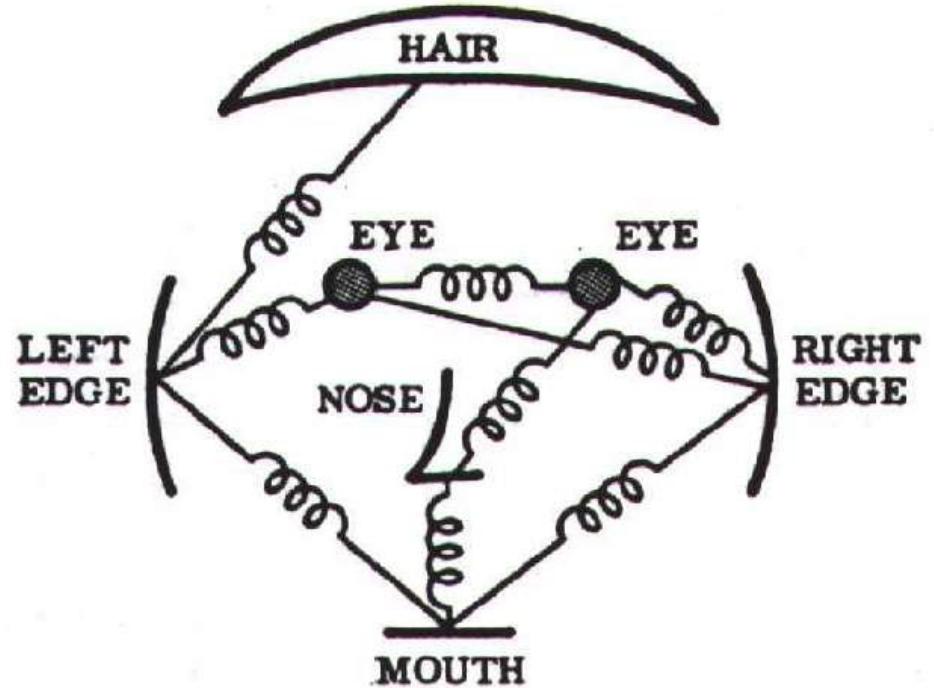
binary images segmented
using skin colour

Implicit Shape Model (ISM)

Representation:

has two components

- parts
(2D image fragments)
- structure
(configuration of parts)

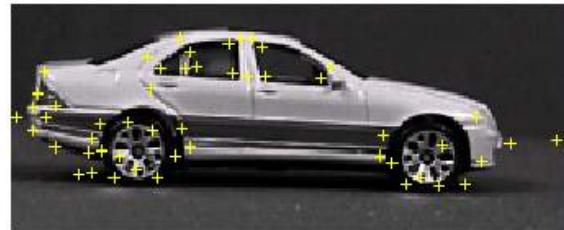


Implicit Shape Model (ISM)

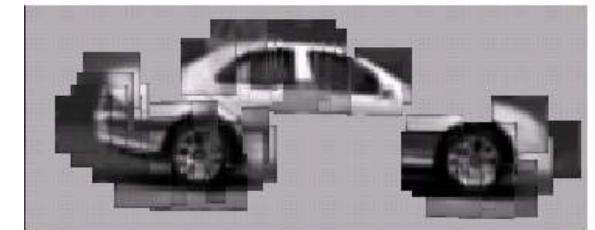
Extraction of local object features:



Training image

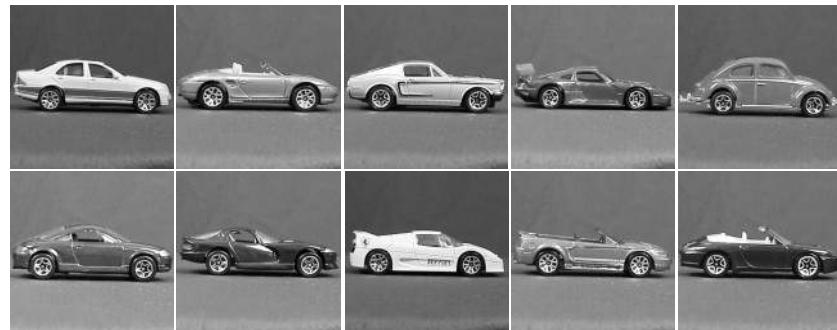


Locate interest
points (e.g. using
Harris detector)



Extract 2D image
patches from
around each
interest point

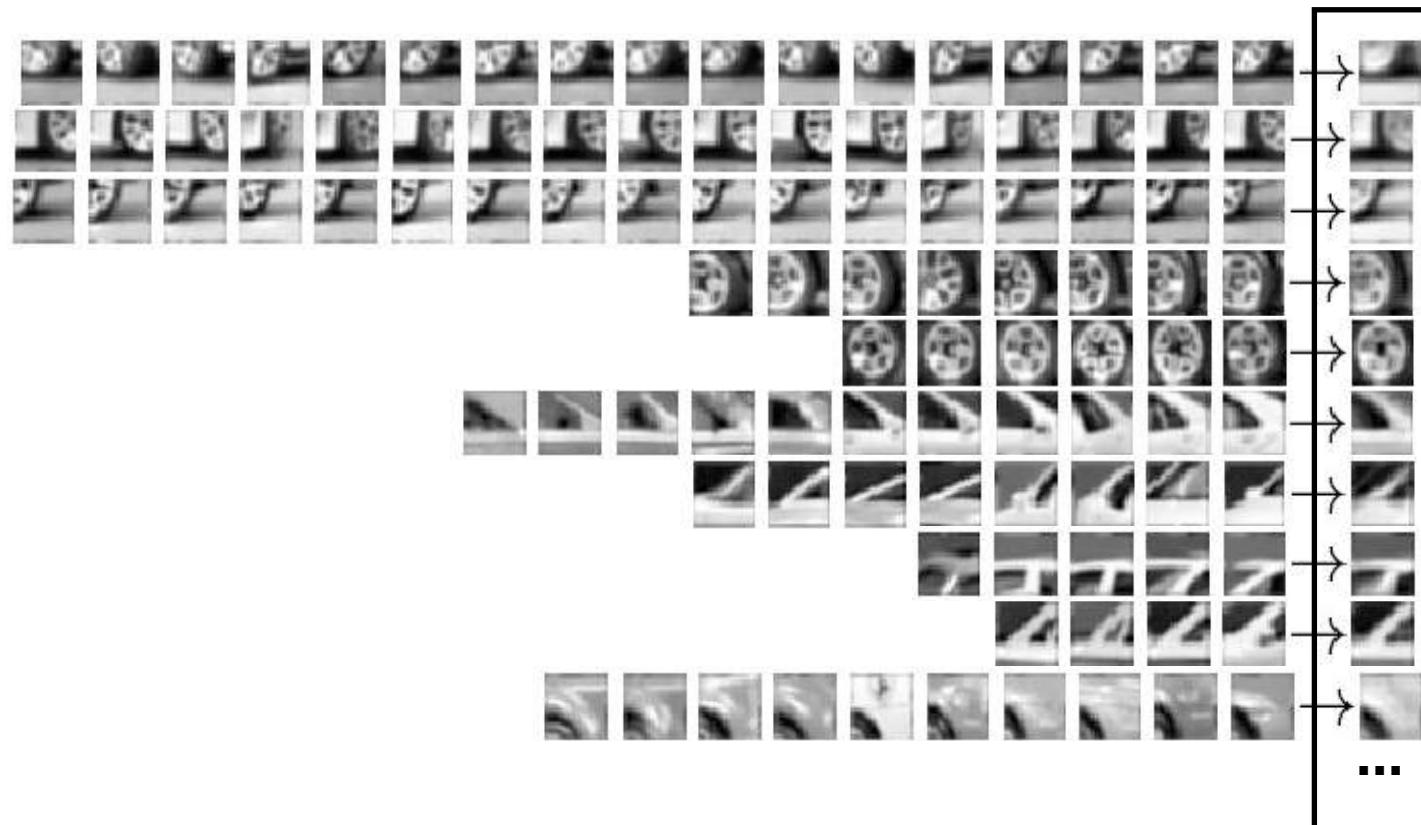
Collect 2D patches (features) from whole training set:



Implicit Shape Model (ISM)

Create appearance codebook:

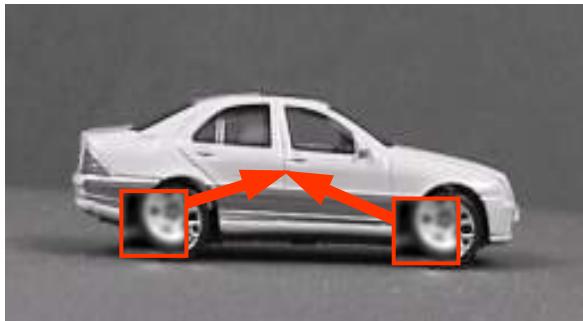
- Cluster patches e.g. using hierarchical agglomerative clustering.
- Store cluster centers as *Appearance Codebook*



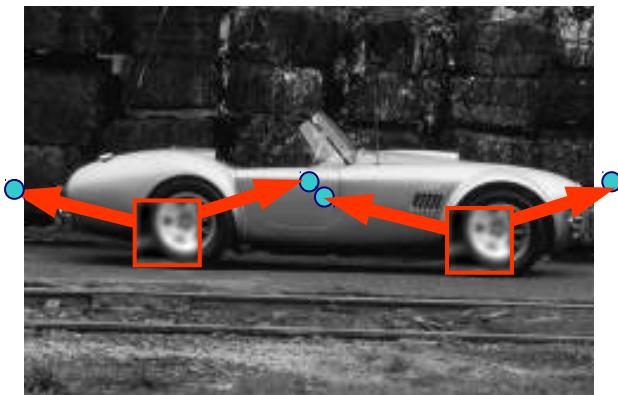
Implicit Shape Model (ISM)

Learn configuration of parts:

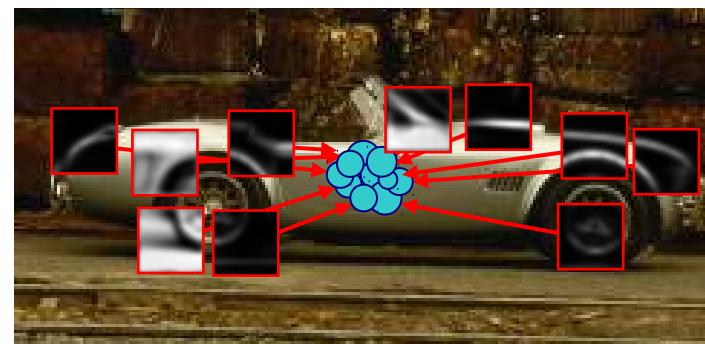
- Match codebook features to training images
- For every codebook entry record possible object centres



During matching procedure, each features votes for possible object centres (equivalent to the Generalized Hough Transform):



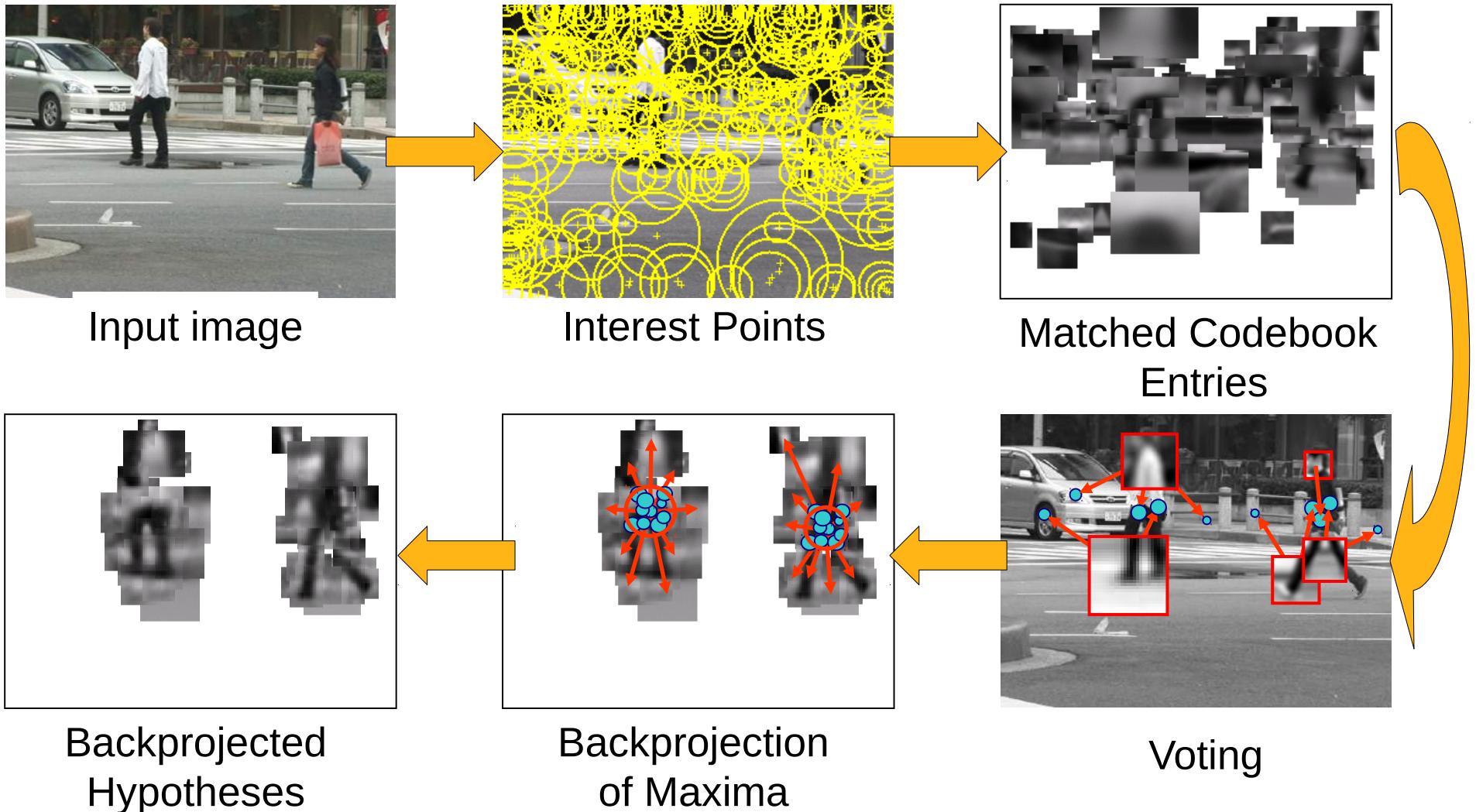
one feature



many features

Implicit Shape Model (ISM)

Matching:

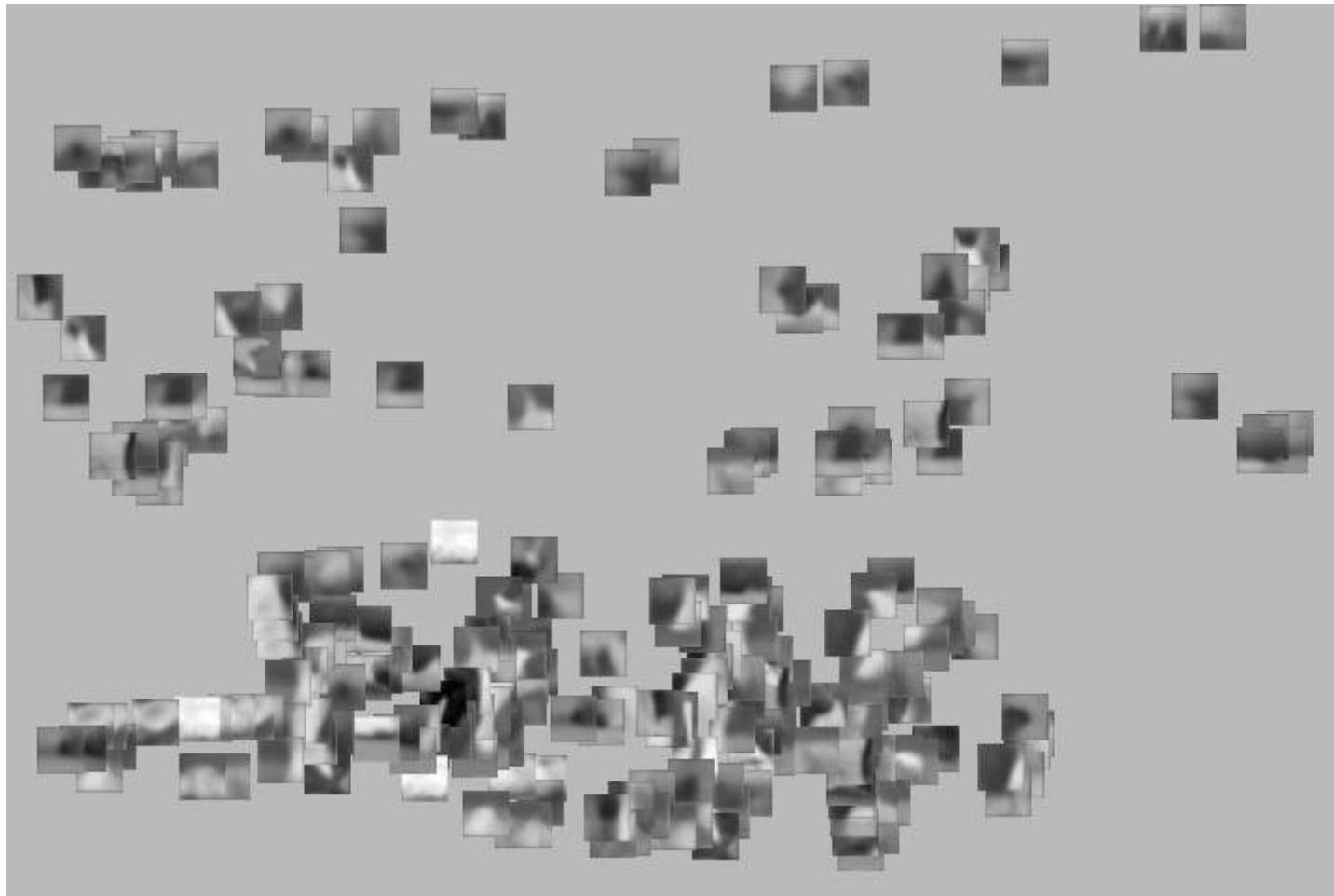


Implicit Shape Model (ISM): example



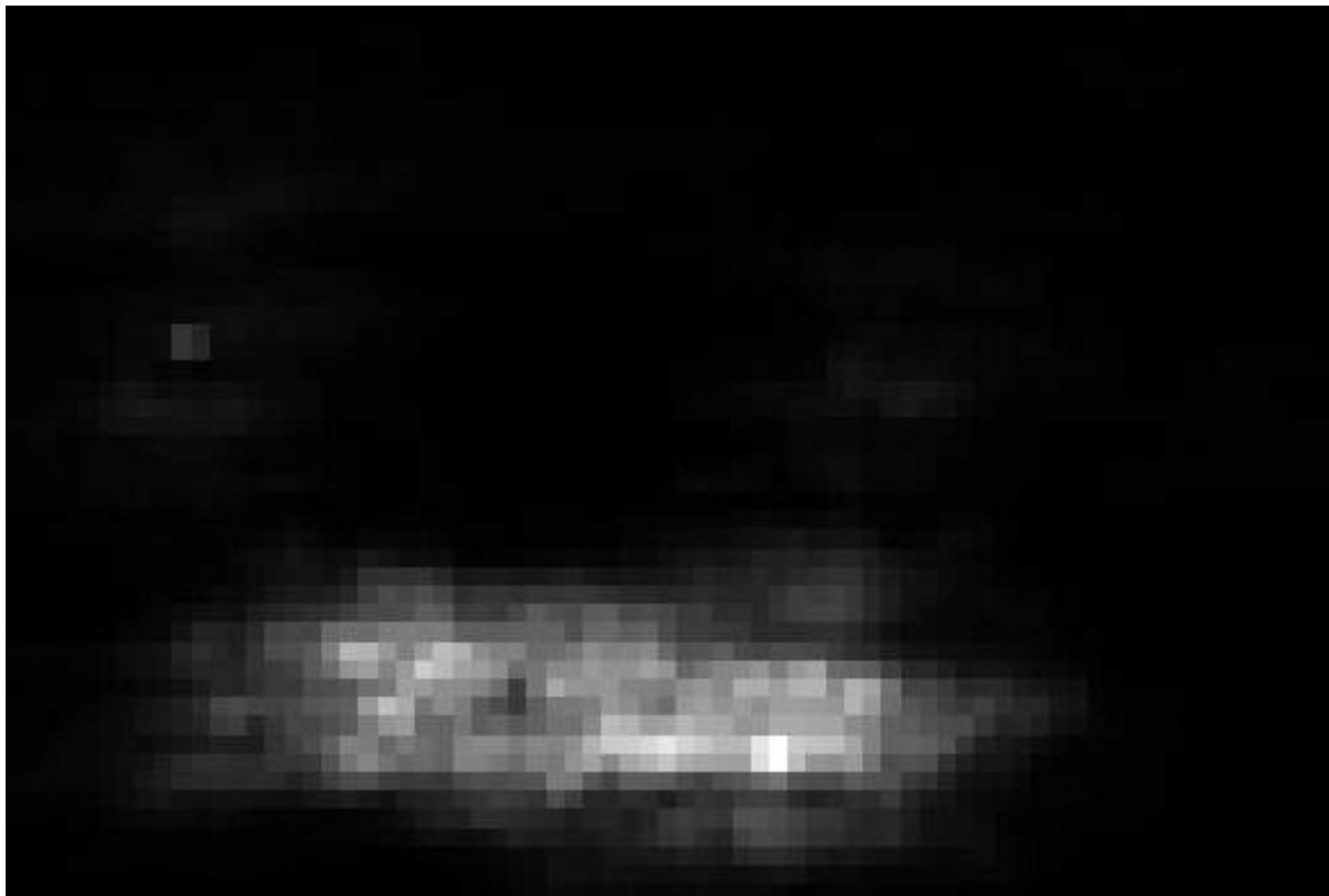
Interest points

Implicit Shape Model (ISM): example



Matched patches

Implicit Shape Model (ISM): example



Votes

Implicit Shape Model (ISM): example



1st hypothesis

Implicit Shape Model (ISM): example



2nd hypothesis

Implicit Shape Model (ISM): example

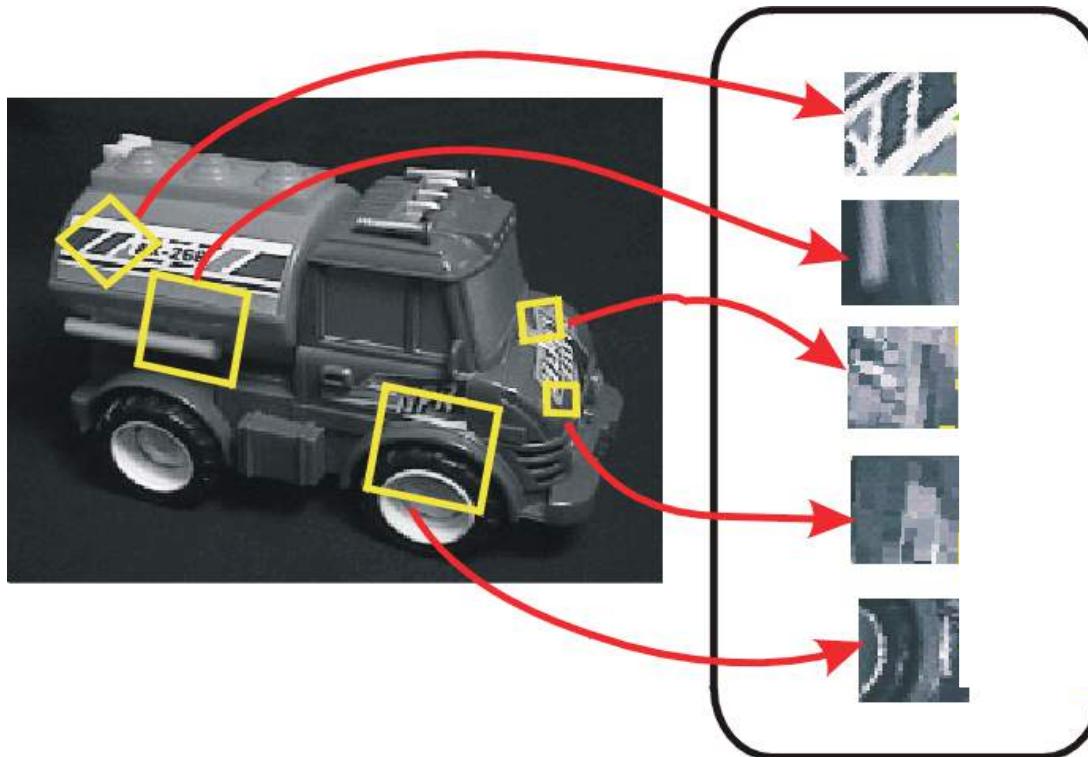


3rd hypothesis

Feature-based object recognition

Representation:

Training image content is transformed into local features that are invariant to translation, rotation, and scale



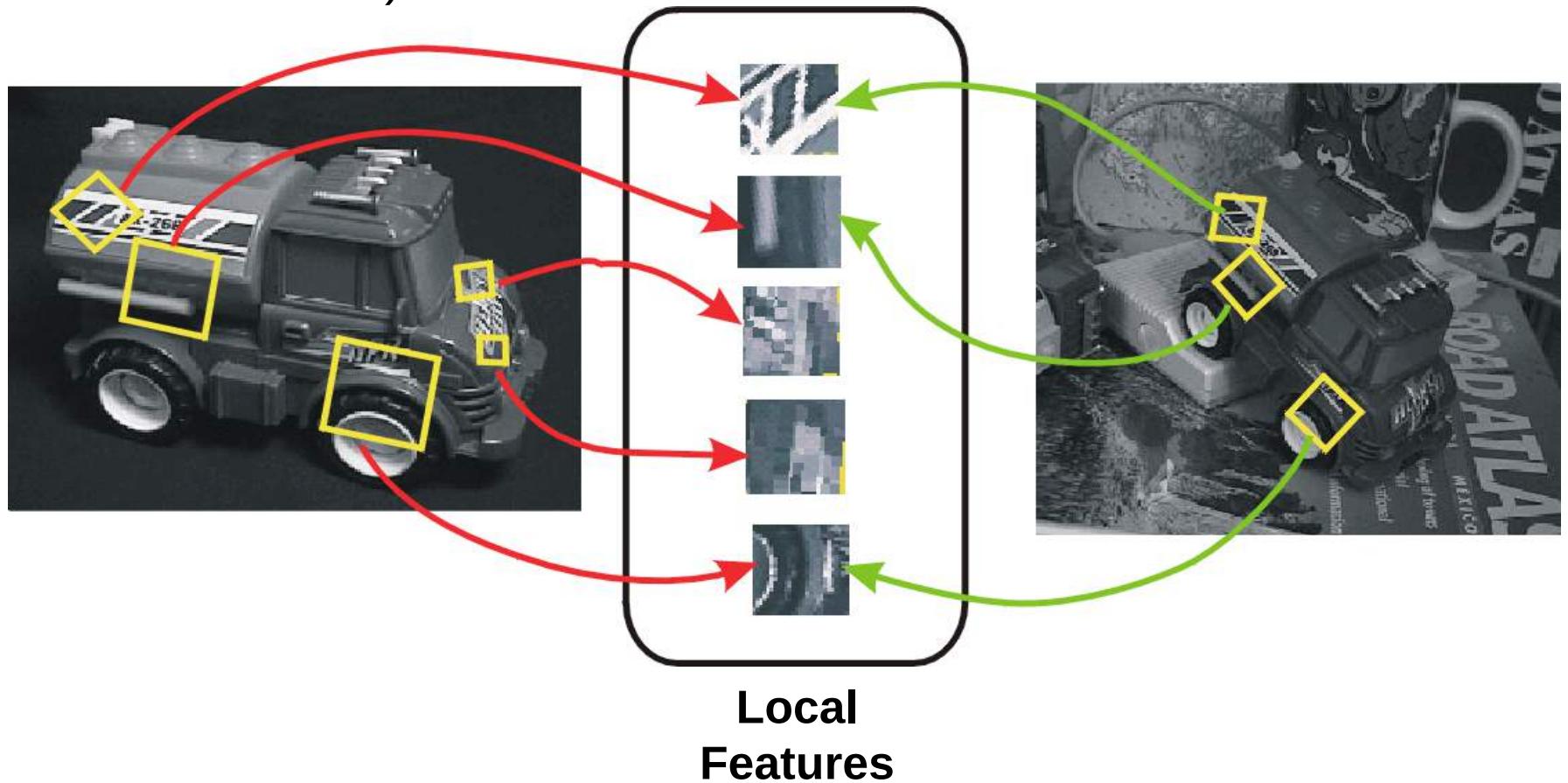
Local Features

Could be patches, but more likely to be descriptors
(like SIFT feature vectors)

Feature-based object recognition

Matching:

Local features are extracted from new image in same way, and matched to those from training image (object recognized if there are sufficient matches)



Feature-based object recognition

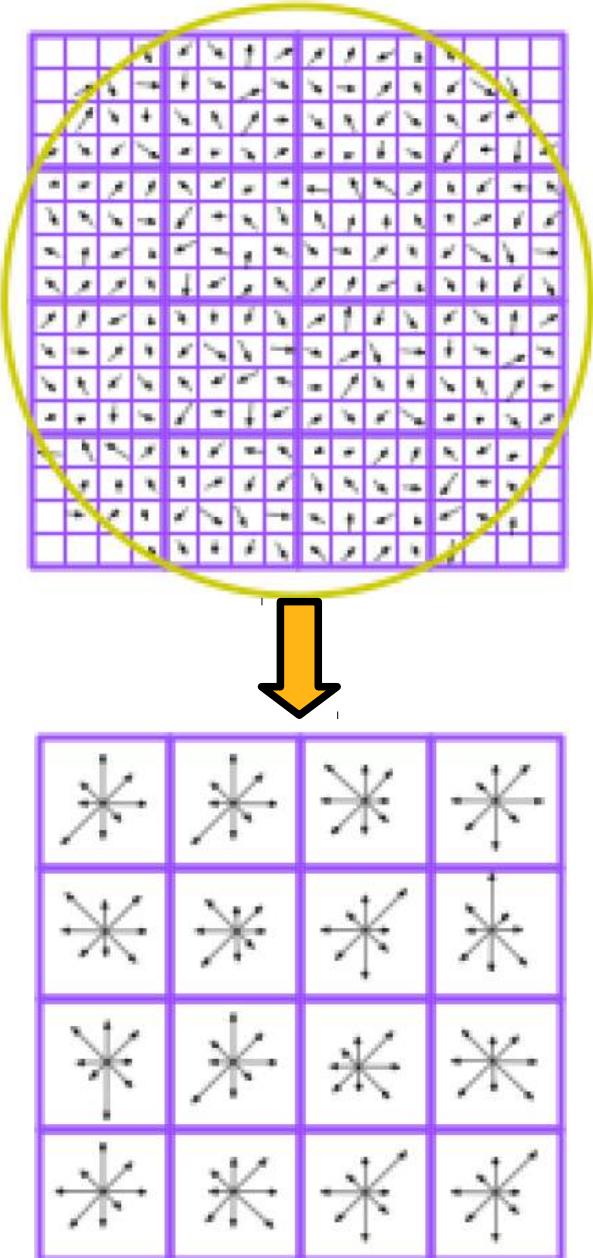
- Feature detection needs to be repeatable despite:
 - Translation, rotation, scale changes
 - Lighting variations
 - Feature detection needs to find sufficient features to cover the object
 - e.g. Harris corner detector, SIFT interest point detector.
 - The features should contain “interesting” structure that can be reliably matched
 - The feature description should be invariant to:
 - Translation, rotation, scale changes
 - Lighting variations
- e.g. SIFT descriptor, or one of many others that have been proposed

SIFT feature matching

Representation:

A 128 element histogram of the orientations of the intensity gradients (binned into 8 orientations) in 4x4 pixels windows around the interest point, normalized so that vector has unit length and rotated so that dominant orientation is vertical.

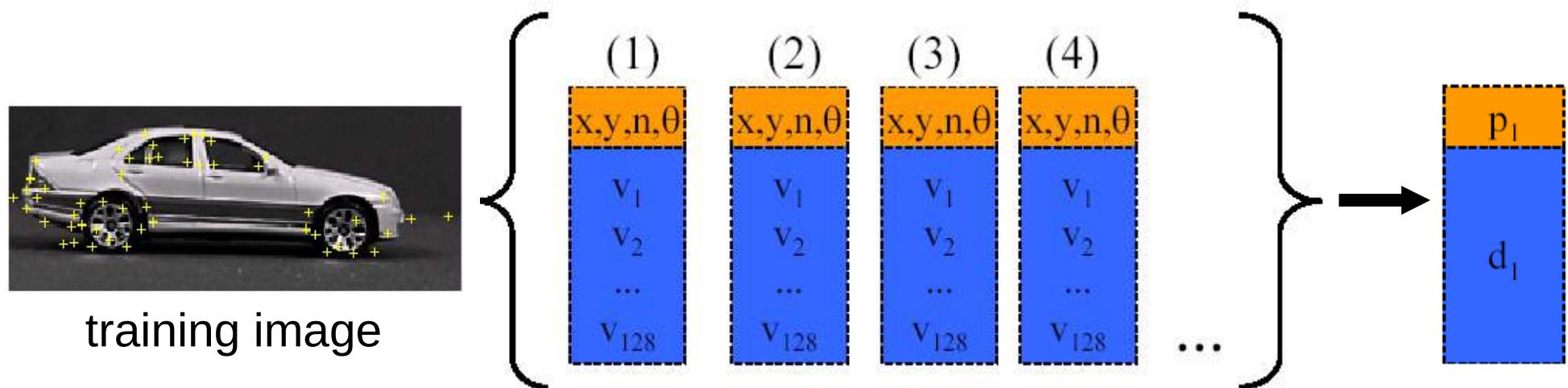
- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance



SIFT feature matching

Representation:

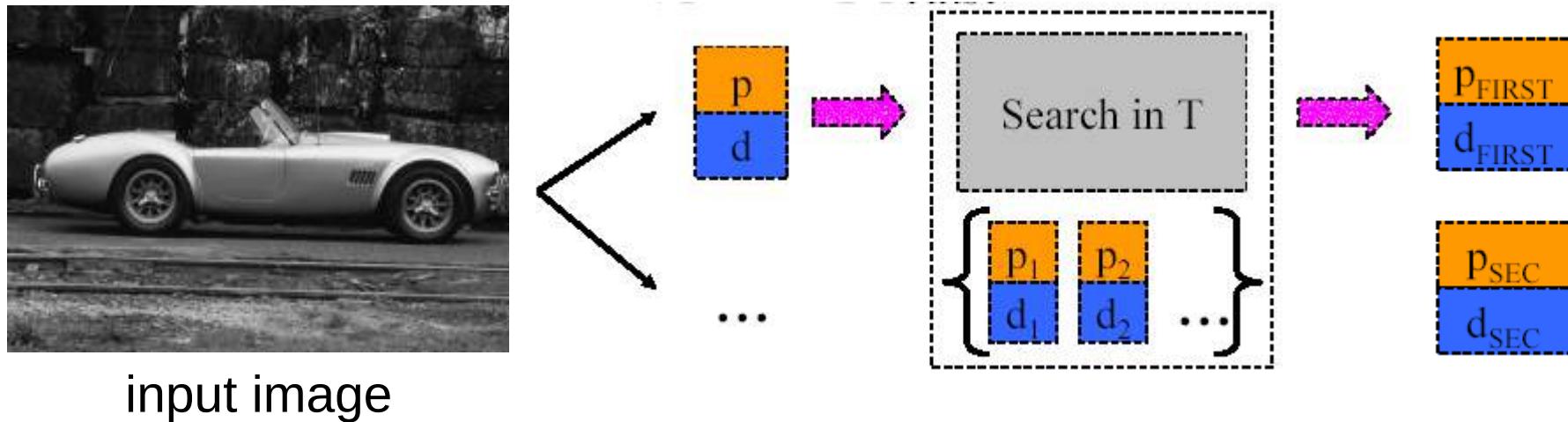
- A set of keypoints are obtained from each training image
- Each such keypoint has a descriptor which is a 128 components vector
- All (keypoint location, image number, feature vector) sets are stored in a database



SIFT feature matching

Matching:

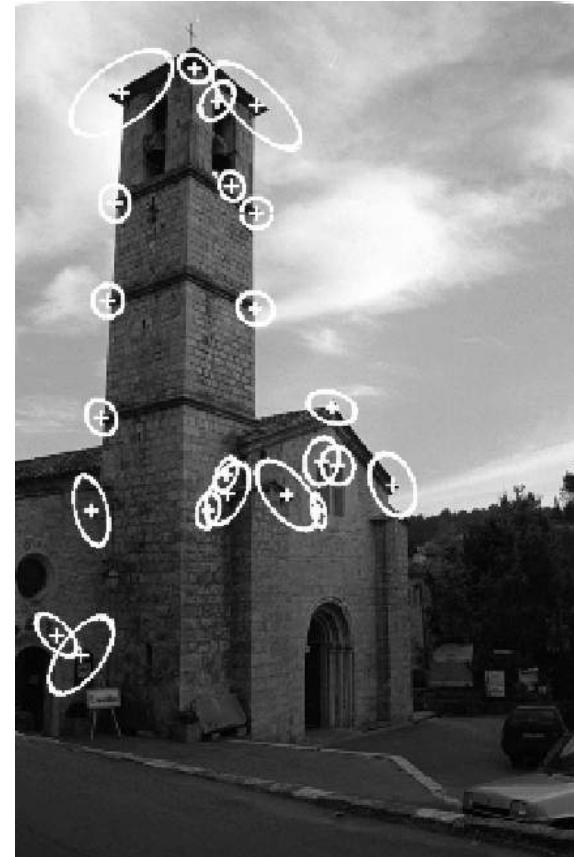
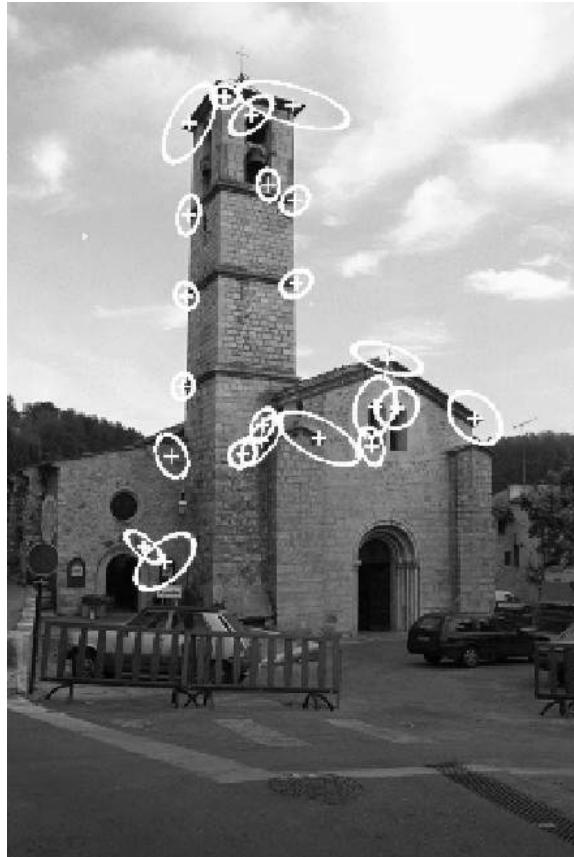
- An input image gives a new set of (keypoint, vector) pairs
- For each such pair, find the top 2 best matching descriptors in the training database



- Match accepted IF
 - Ratio of distance to first nearest descriptor to that of second < threshold

SIFT feature matching: example

Recognition with changes in viewpoint and illumination

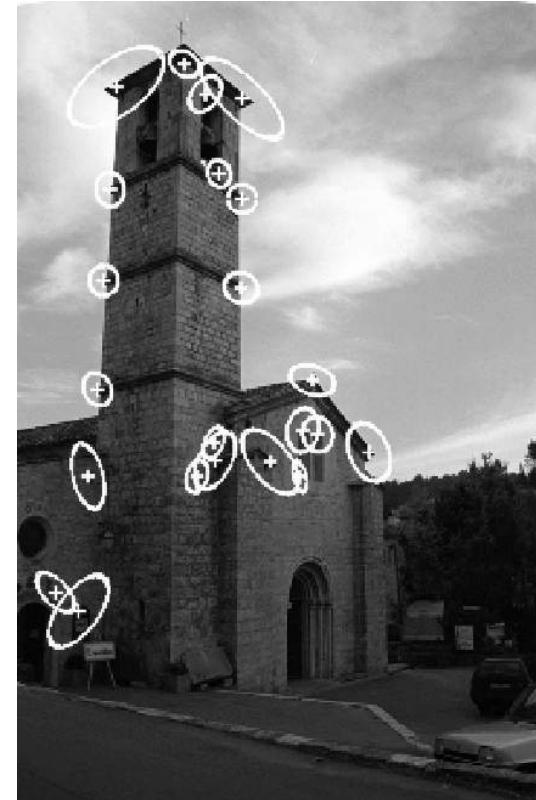
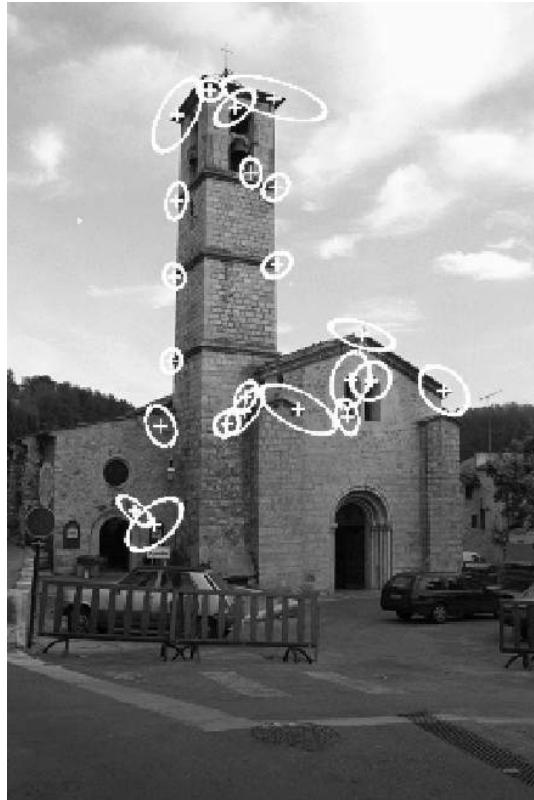


22 correct matches

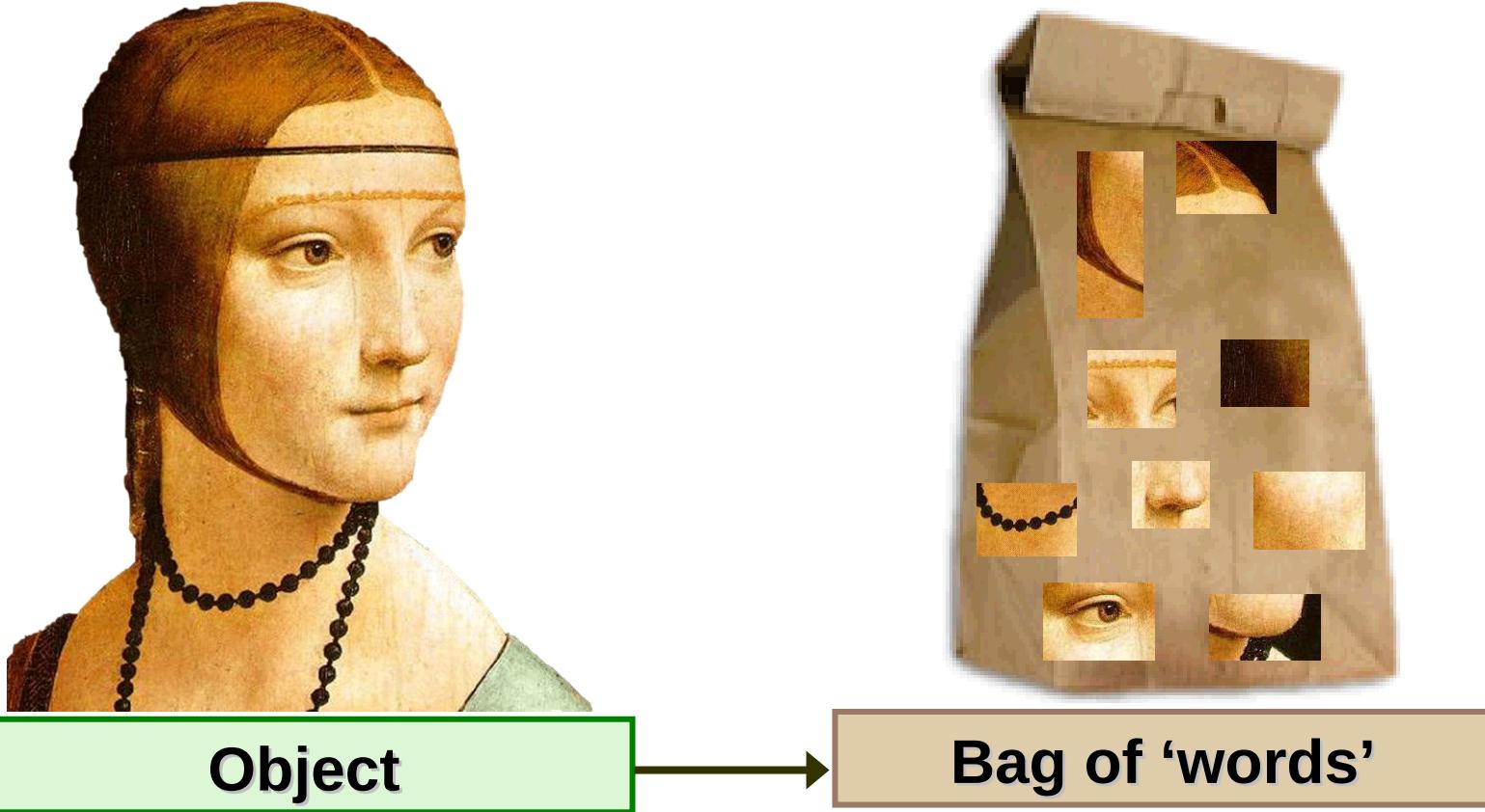
SIFT feature matching: confirmation

Given three non-collinear model points P_1, P_2, P_3 , and three image points p_1, p_2, p_3 , there is a unique transformation (rotation, translation, scale) that aligns the model with the image.

Use RANSAC or Generalised Hough Transform to determine if matched locations are consistent (i.e. can be modelled by the same transformation from one view to another).



Bag-of-words



Analogous to the method used for document recognition by Google.

Bag-of-words (for documents)

Representation:

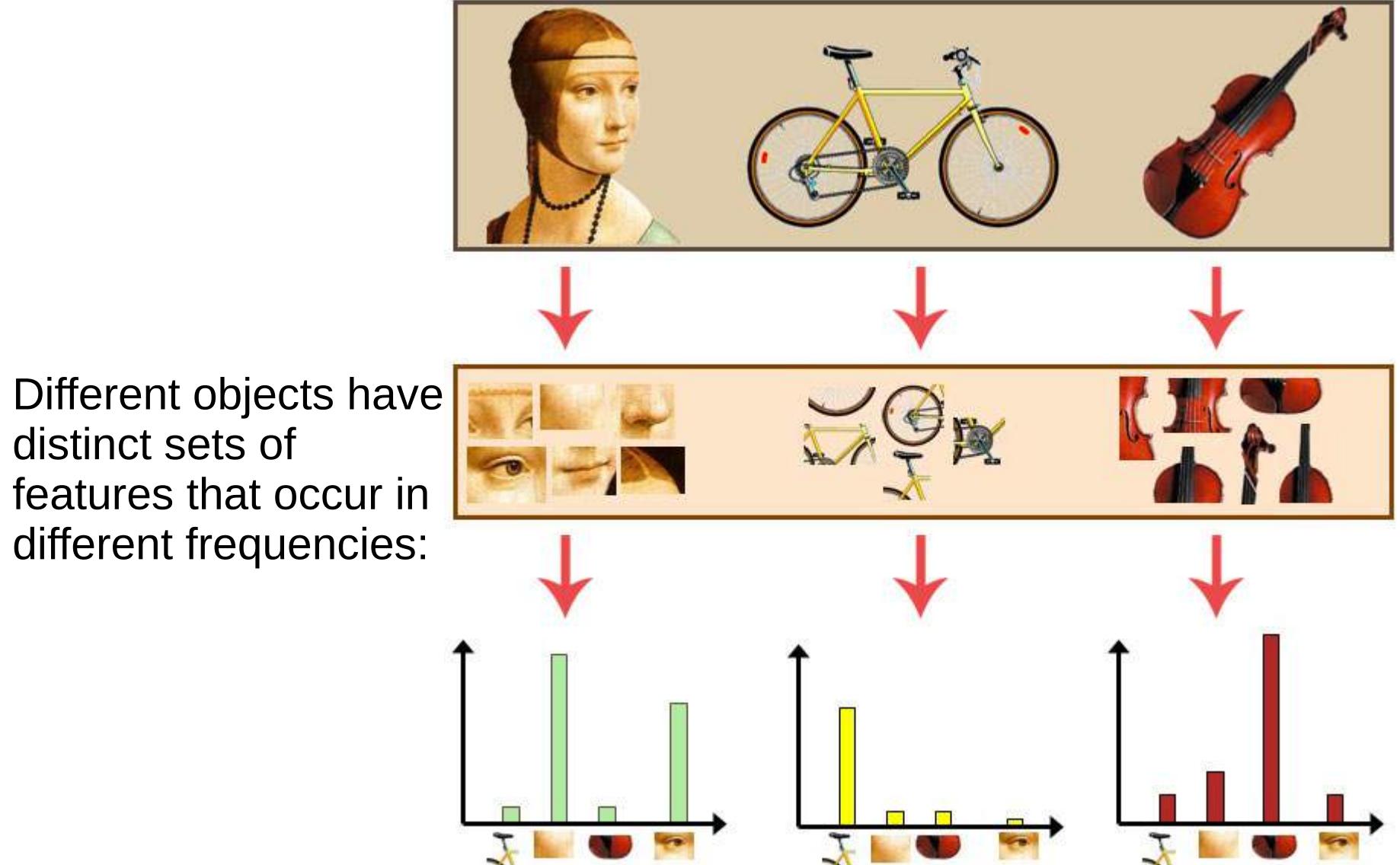
- Documents are parsed into words
 - e.g. “the cat sat on the mat, a dog is sitting on the cat”
- Common words are ignored (the, a, on, it, he, etc.)
 - e.g. “cat sat mat dog sitting cat”
- Words are represented by their stems (e.g. ‘sitting’, ‘sat’, ‘sits’ all become ‘sit’)
 - e.g. “cat sit mat dog sit cat”
- Each word is assigned a unique identifier
 - e.g. 1 = “cat”, 2= “sit”, 3= “mat” 4= “dog” 5=“fish”
- Each document is represented by a K components vector of words frequencies (where K is the total size of the vocabulary extracted from all documents)
 - e.g. (2, 2, 1, 1, 0, ...)

Bag-of-words (for documents)

Matching:

- The query is represented in the same format.
 - e.g. “cats and dogs” → (1, 0, 0, 1, ...)
- For all documents containing at least one of the query words, calculate the angle (i.e. $\cos^{-1}NCC$) between the query and document vectors
- Rank the results

Bag-of-words (for images)



Different objects have distinct sets of features that occur in different frequencies:

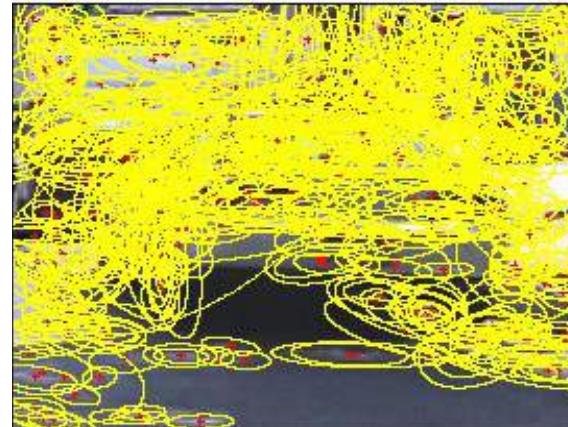
Represent objects as a distribution (histogram) of feature occurrences.

Bag-of-words

Choosing features:



Regular grid



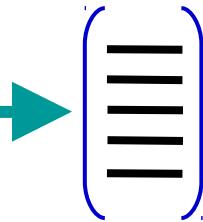
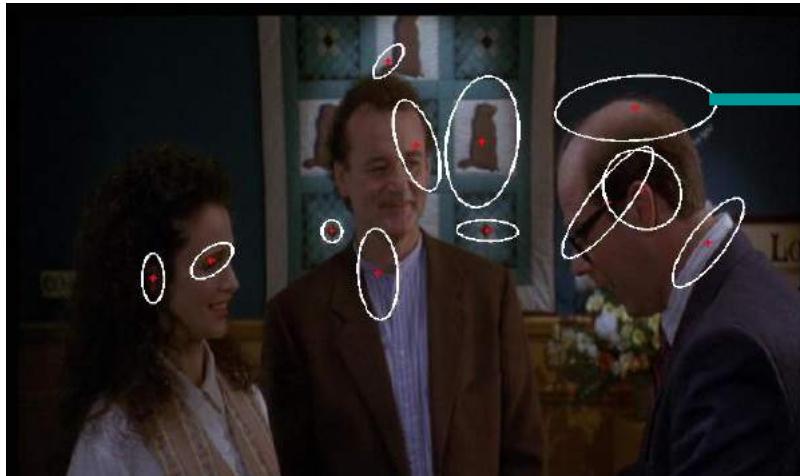
Interest point detector



Randomly

Bag-of-words

Encoding features:

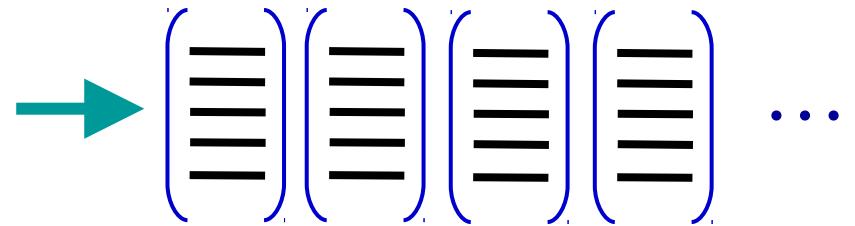
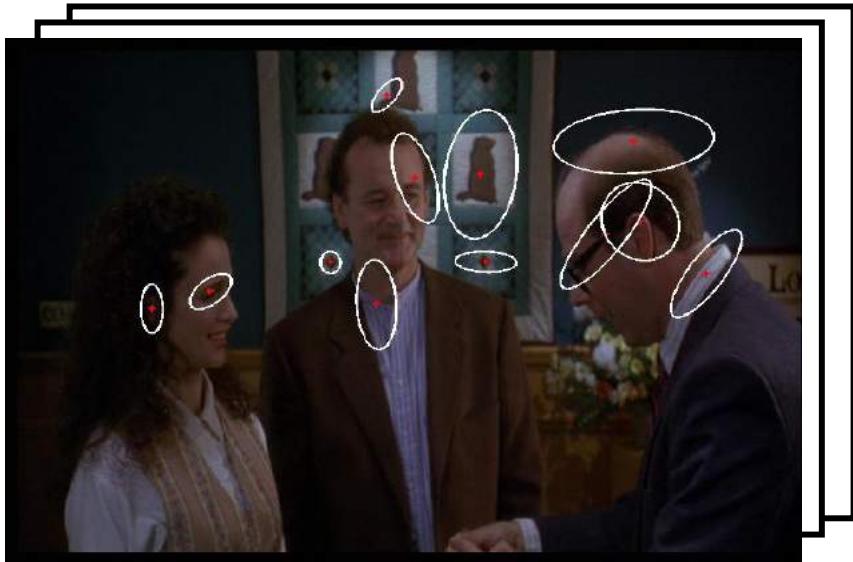


Compute
SIFT
descriptor

Image patches around the chosen locations are encoded using descriptor
(analogous to a word)

Bag-of-words

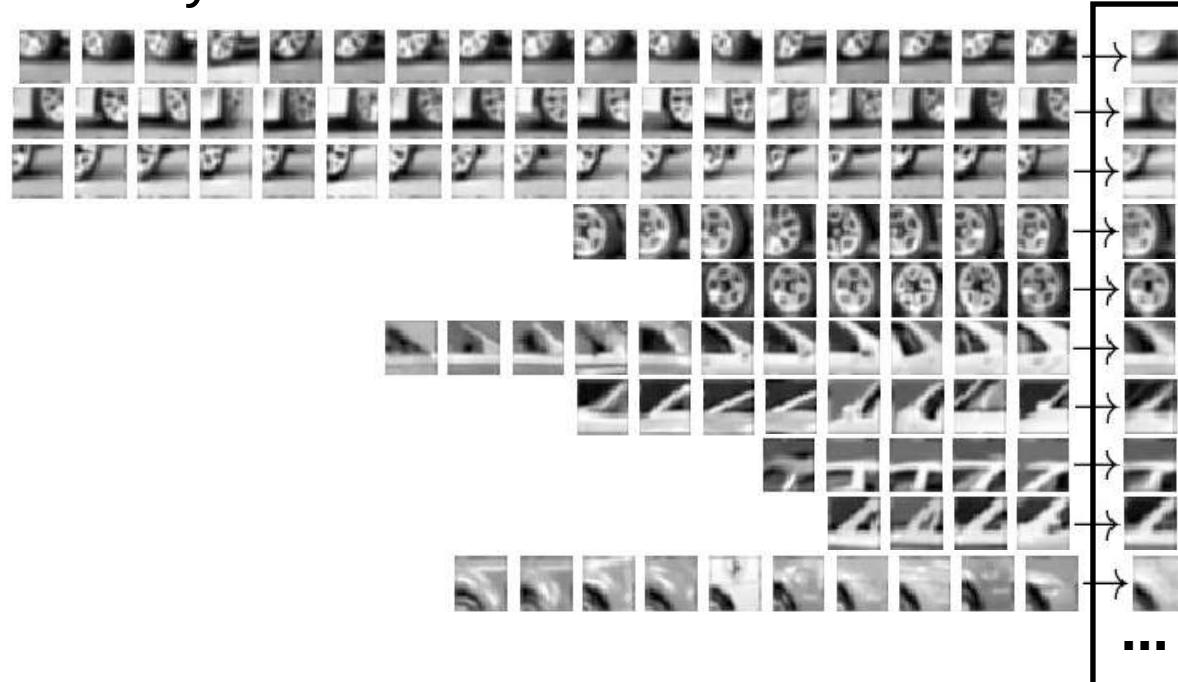
Creating dictionary:



Encode many features taken from many images

Bag-of-words

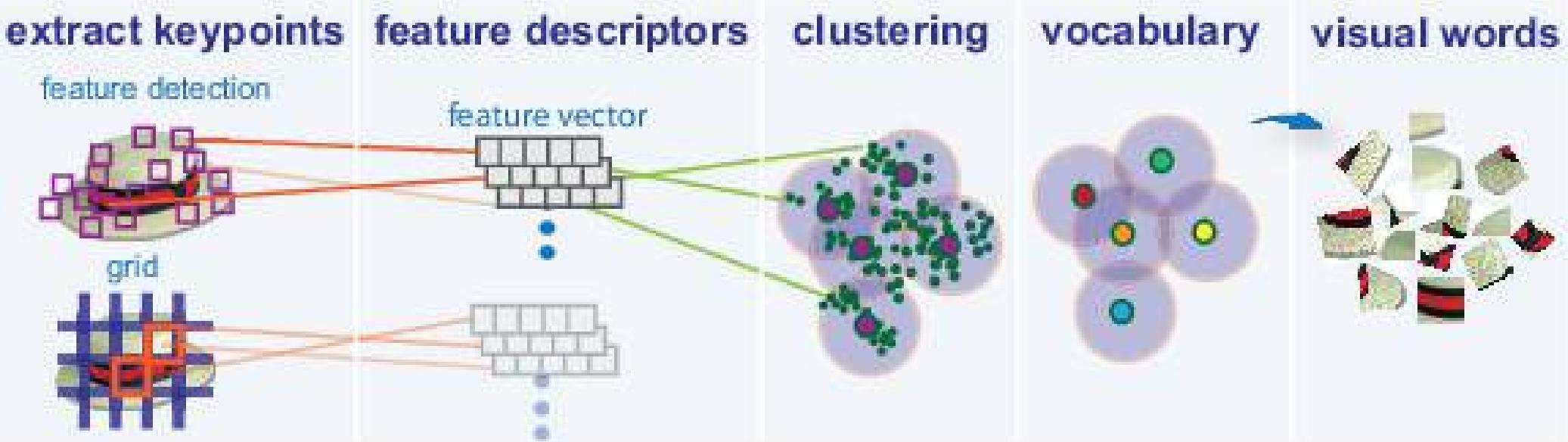
Creating dictionary:



- cluster feature descriptors into K groups using K-means clustering algorithm (analogous to representing words by their stems)
- each cluster represents a “visual word” or codeword.
- the whole set of clusters represents a “visual vocabulary” or codeword dictionary
- The most frequent codewords that occur in almost all images are removed from the dictionary (analogous to ignoring common words)

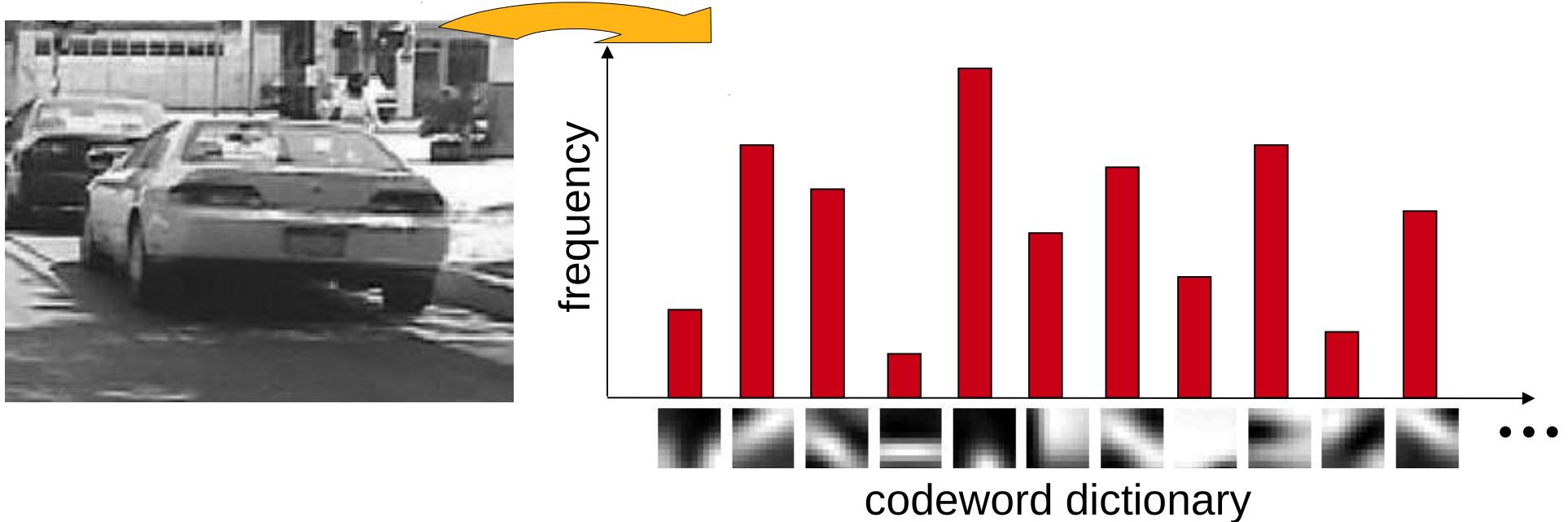
Bag-of-words

Creating dictionary (summary):



Bag-of-words

Representing images:



Representation:

Each image is represented as a histogram showing the frequency of appearance of each of the codewords in the dictionary.

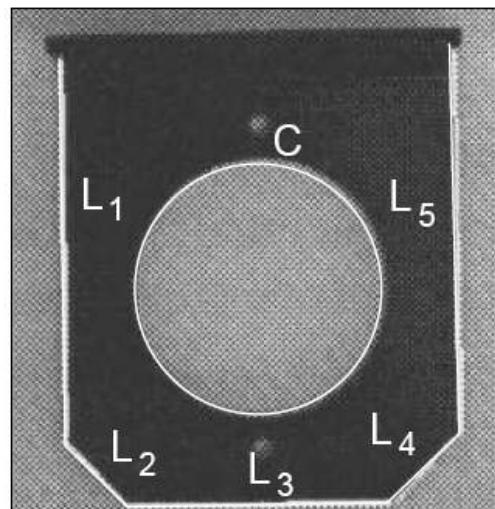
Matching:

Images compared (and hence a match between an input image and a training image) is found by calculating the distance between histograms.

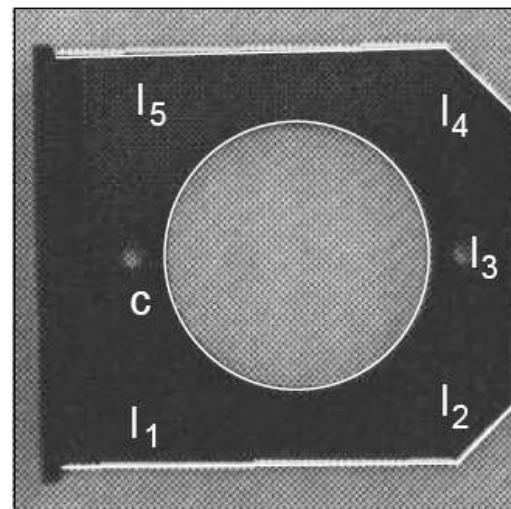
Geometric invariants

A geometric invariant is a property of an object in the scene, which does not vary with viewpoint.

e.g. if we consider viewpoint changes in Euclidean space (i.e. translation and rotation)



Euclidean
→

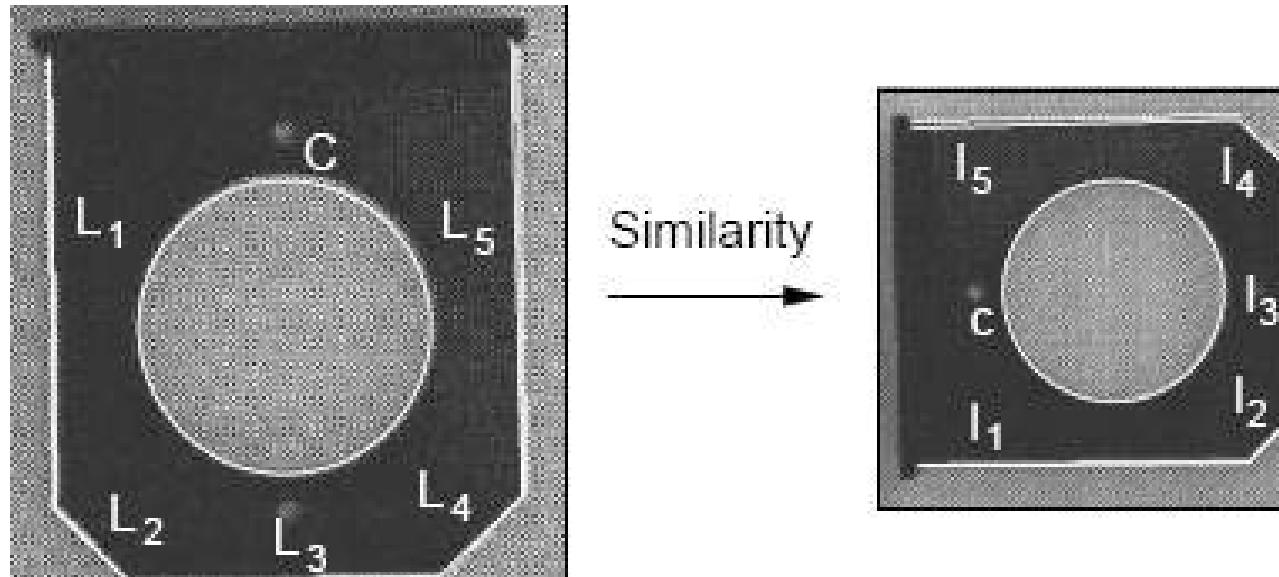


invariant properties:

- lengths
- angles
- areas

Geometric invariants

e.g. if we consider viewpoint changes in Similarity space (i.e. translation, rotation and scale)

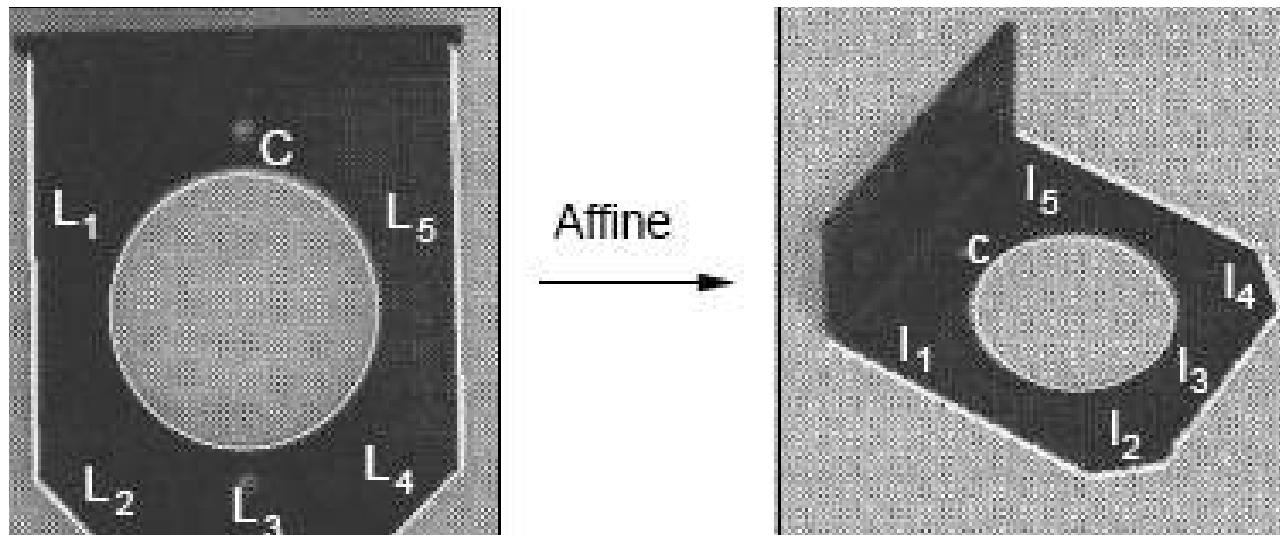


invariant properties:

- *ratios of lengths*
- *angles*

Geometric invariants

e.g. if we consider viewpoint changes in Affine space (i.e. translation, rotation, scale and shear)

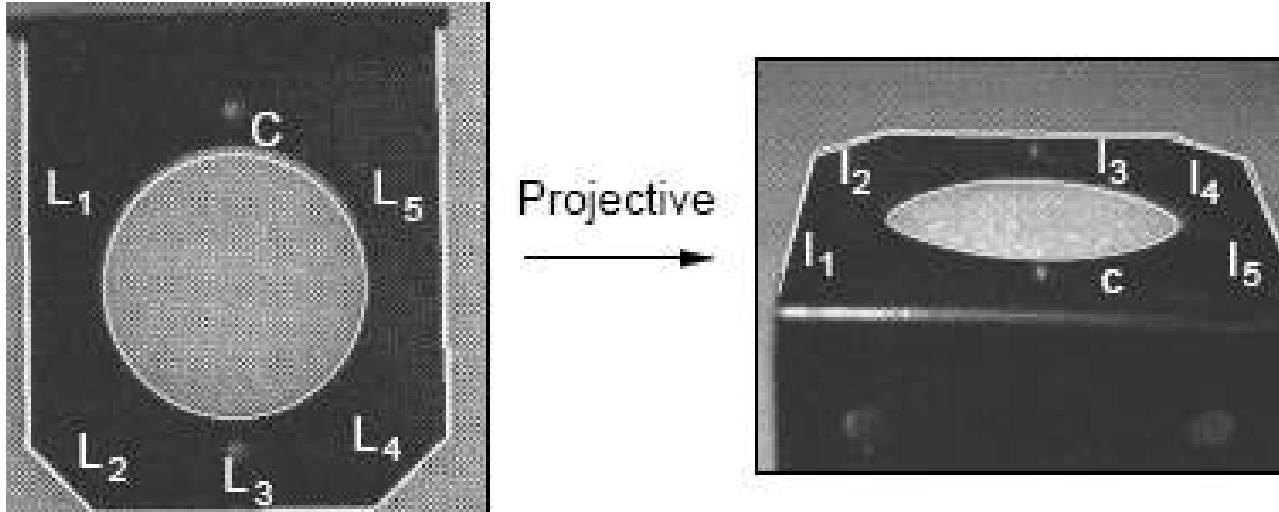


invariant properties:

- parallelism
- *ratios of lengths along lines*
- *ratio of areas*

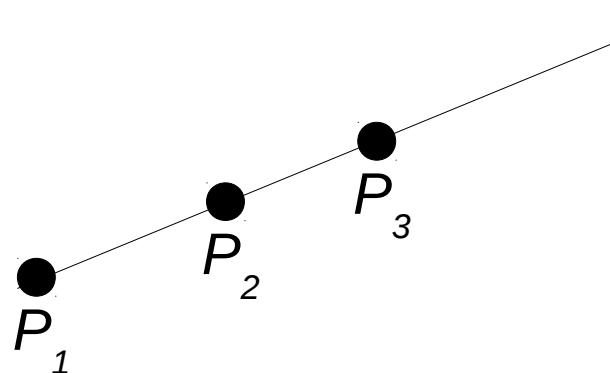
Geometric invariants

e.g. if we consider viewpoint changes in Projective space (i.e. translation, rotation, scale, shear and foreshortening)



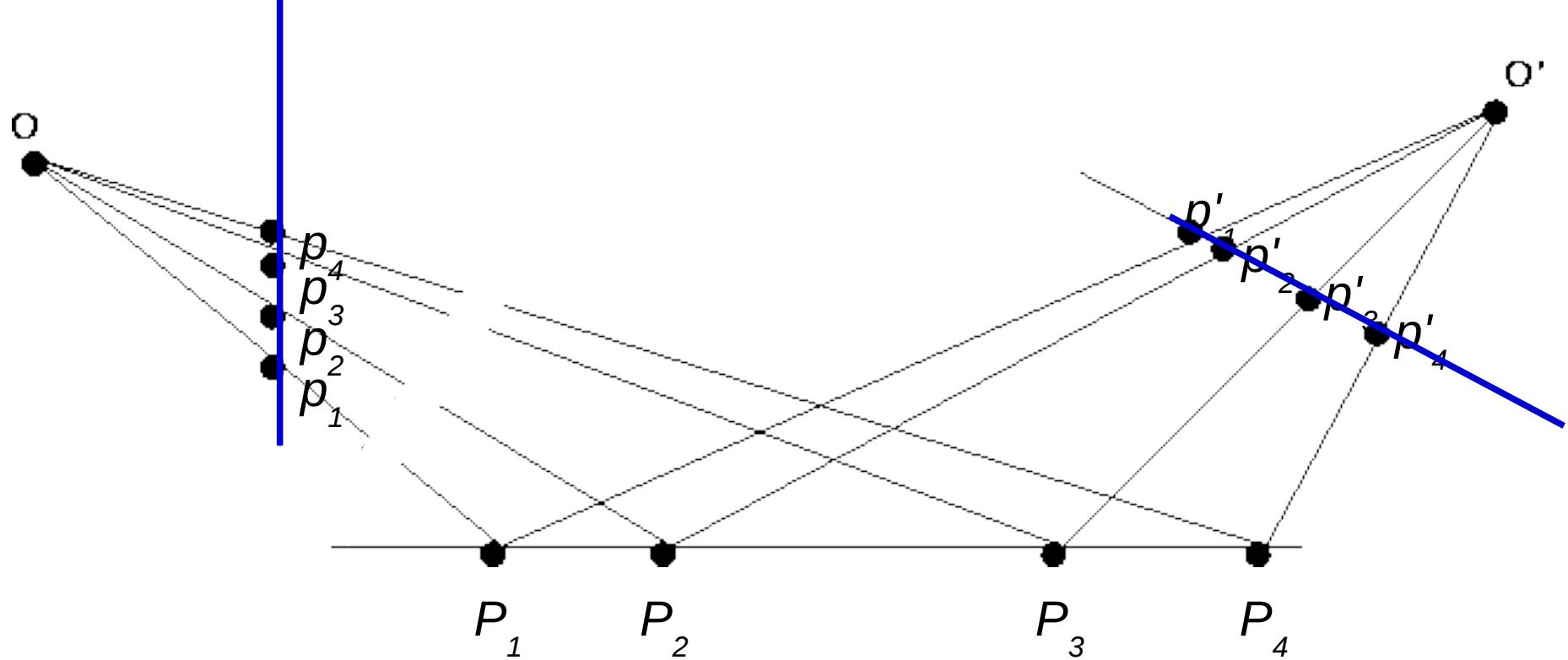
invariant properties:
• *cross-ratio*

The *cross-ratio* is the ratio of ratios of lengths on a line, e.g.



$$\frac{\|P_3 - P_1\| \|P_4 - P_2\|}{\|P_3 - P_2\| \|P_4 - P_1\|}$$

Geometric invariants



$$\frac{\|P_3 - P_1\| \|P_4 - P_2\|}{\|P_3 - P_2\| \|P_4 - P_1\|} = \frac{\|p_3 - p_1\| \|p_4 - p_2\|}{\|p_3 - p_2\| \|p_4 - p_1\|} = \frac{\|p'_3 - p'_1\| \|p'_4 - p'_2\|}{\|p'_3 - p'_2\| \|p'_4 - p'_1\|}$$

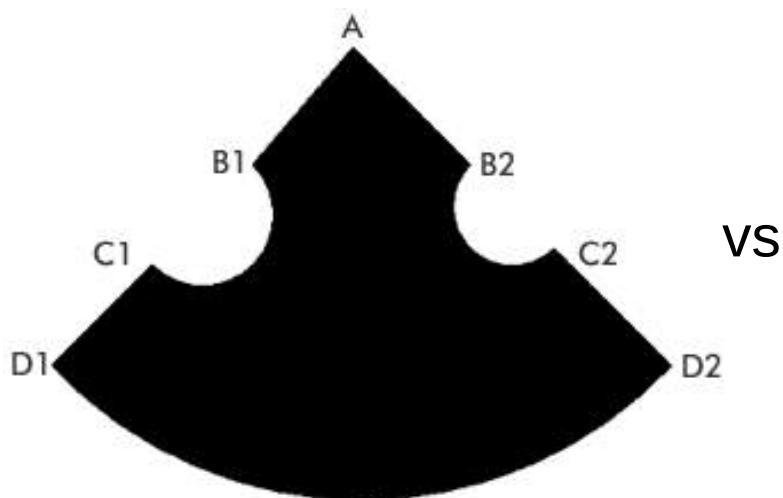
Under perspective projection, the cross ratio remains constant from any viewpoint.

Geometric invariants

Representation: value of cross-ratio

Matching: compare value of cross-ratio measured in image with database of cross-ratios measured in training images.

Problems: occlusion, availability and distinctiveness of the points,
e.g.:



VS



Summary

- template matching
- *sliding window*
- edge matching
- model-based
- intensity histograms
- *implicit shape model*
- *SIFT feature matching*
- *bag-of-words*
- geometric invariants

These are just a small selection of methods – many, many, more methods have been tried!

There are various ways of classifying methods...

Classification of Object Recognition Methods

Matching procedure:

- Top-down (generative)

e.g. model-based, templates, edge matching

Hypothesise that image contains a certain object and look for it in the image. Expectation-driven. (= “fitting”, see lecture on segmentation).

- Bottom-up (discriminative)

e.g. SIFT, bag-of-words, intensity histograms, ISM

Extract description and match to descriptions in database. Stimulus-driven.

Classification of Object Recognition Methods

Representation used:

- pixel intensities vs feature vectors vs geometry
 - e.g. templates, edge matching, intensity histograms, ISM
 - e.g. SIFT feature matching, bag-of-words
 - e.g. model-based, geometric invariants
- 2D (image-based) vs 3D (object-based)
 - e.g. templates, sliding window, ISM, bag-of-words
 - e.g. 3D models, geometric invariants, SIFT (check of consistency)
- local features vs global features
 - e.g. SIFT (bag-of-words), ISM, part templates
 - e.g. whole object templates, sliding window, 3D models

Local vs global representation

Local (each object is broken down into simple features)

e.g. $A = / + \backslash + -$

Advantages: tolerant to viewpoint, within class variation, occlusion.

Problem: Many objects consist of the same collection of features, and hence can not be distinguished

e.g. $T = | + -$

$L = | + -$

$+ = | + -$

Problem: If many objects in image, then there is no information about which feature comes from which object

e.g. $TA = | + - + / + \backslash + -$

$IV = | + - + / + \backslash + -$

Local vs global representation

Global (each object is represented by a description of its overall shape)

e.g. $A = A$

Advantage: can distinguish similar objects

Problems:

- sensitive to viewpoint and within class variation (due to need for exact alignment of representations)
- sensitive to occlusion (due to all parts of description being relevant for matching)

Local vs global representation

Local and global representations have complementary advantages and disadvantages

- local features generate many false positives
- global features generate many false negatives

Solutions:

- use features of intermediate complexity
- use a hierarchy of features with a range of complexities

Computer Vision (7CCSMCVI / 6CCS3COV)

Recap

- **Image formation**
- **Low-level vision**
- **Mid-level vision**
- **High-level vision**
 - **Artificial**
 - template matching
 - sliding window
 - edge matching
 - model-based
 - intensity histograms
 - implicit shape model
 - SIFT feature matching
 - bag-of-words
 - geometric invariants
 - **Biological**

← Today

Today

- Theories of object recognition / categorisation:
 - object-based (3D) vs image-based (2D)
 - configural (global) vs featural (local)
 - rules vs exemplars vs prototypes
- Theories of cortical processing:
 - hierarchical neural network models
 - » Feedforward (HMAX, CNN)
 - » Recurrent
- Top-down vs Bottom-up
 - Bayesian inference

Object based vs Image based theories

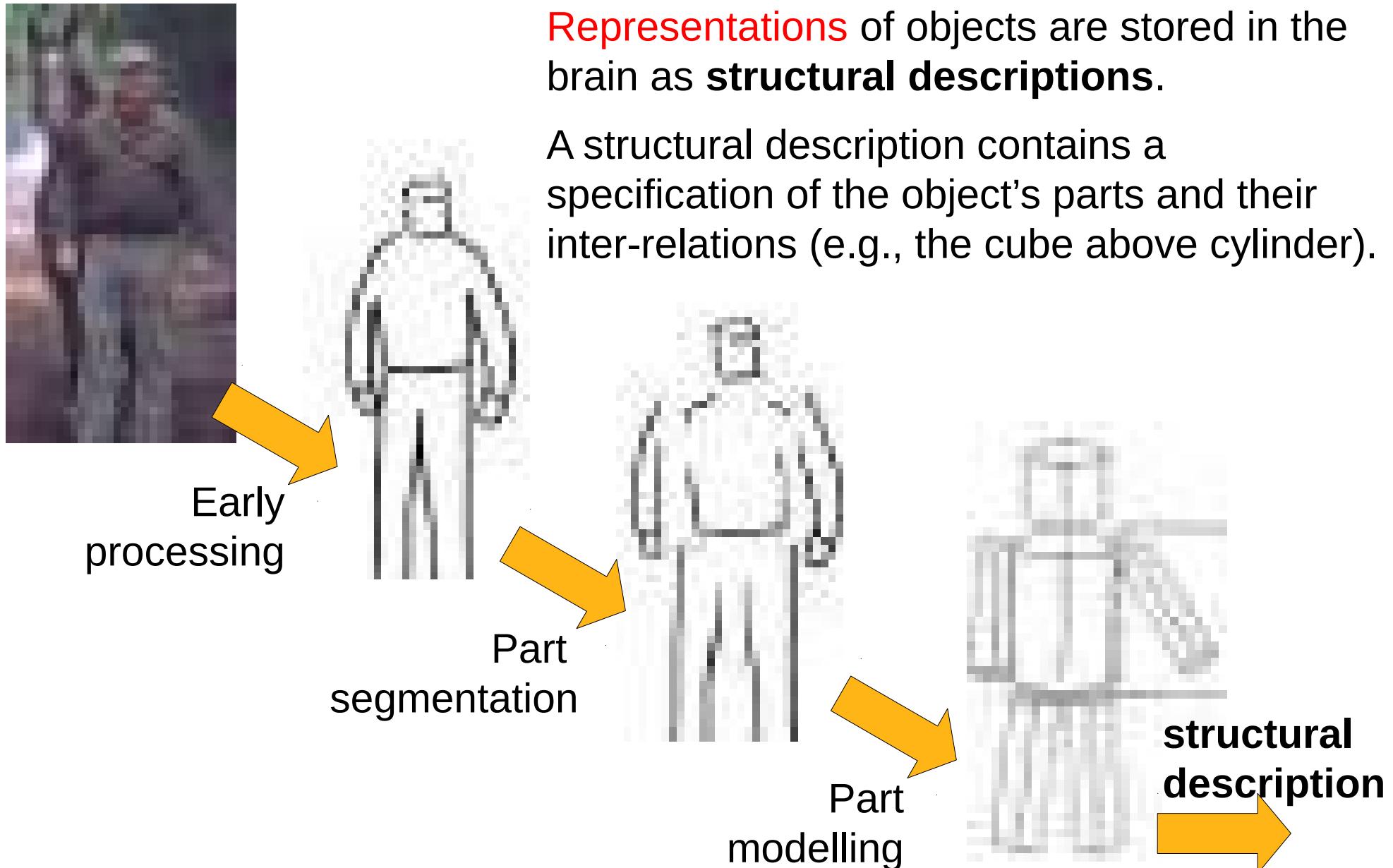
Object based:

- each object represented by storing a 3D model
- object-centred reference frame

Image based:

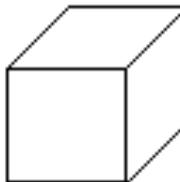
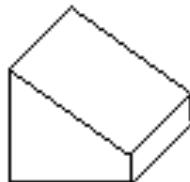
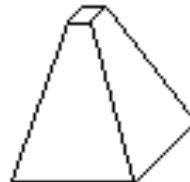
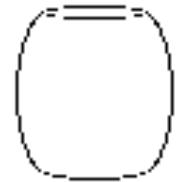
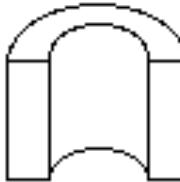
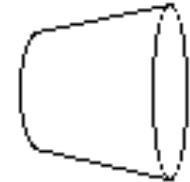
- each object represented by storing multiple 2D views (e.g. images)
- viewer-centred reference frame

Object based: Recognition By Components



Object based: Recognition By Components

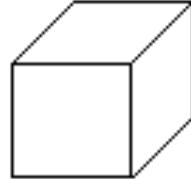
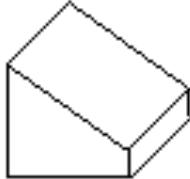
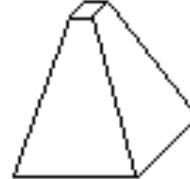
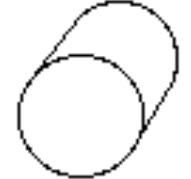
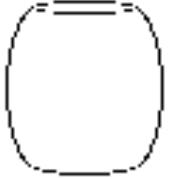
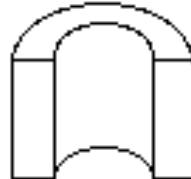
Hypothesis: there is a small number of geometric components that constitute the primitive elements of the object recognition system (like letters forming words).

 Cube Straight Edge Straight Axis Constant	 Wedge Straight Edge Straight Axis Expanded	 Pyramid Straight Edge Straight Axis Expanded	 Cylinder Curved Edge Straight Axis Constant	 Barrel1 Curved Edge Straight Axis Exp & Cont
 Arch Straight Edge Curved Axis Constant	 Cone Curved Edge Straight Axis Expanded	 Expanded Cylinder Curved Edge Straight Axis Expanded	 Handle Curved Edge Curved Axis Constant	 Expanded Handle Curved Edge Curved Axis Expanded

Object based: Recognition By Components

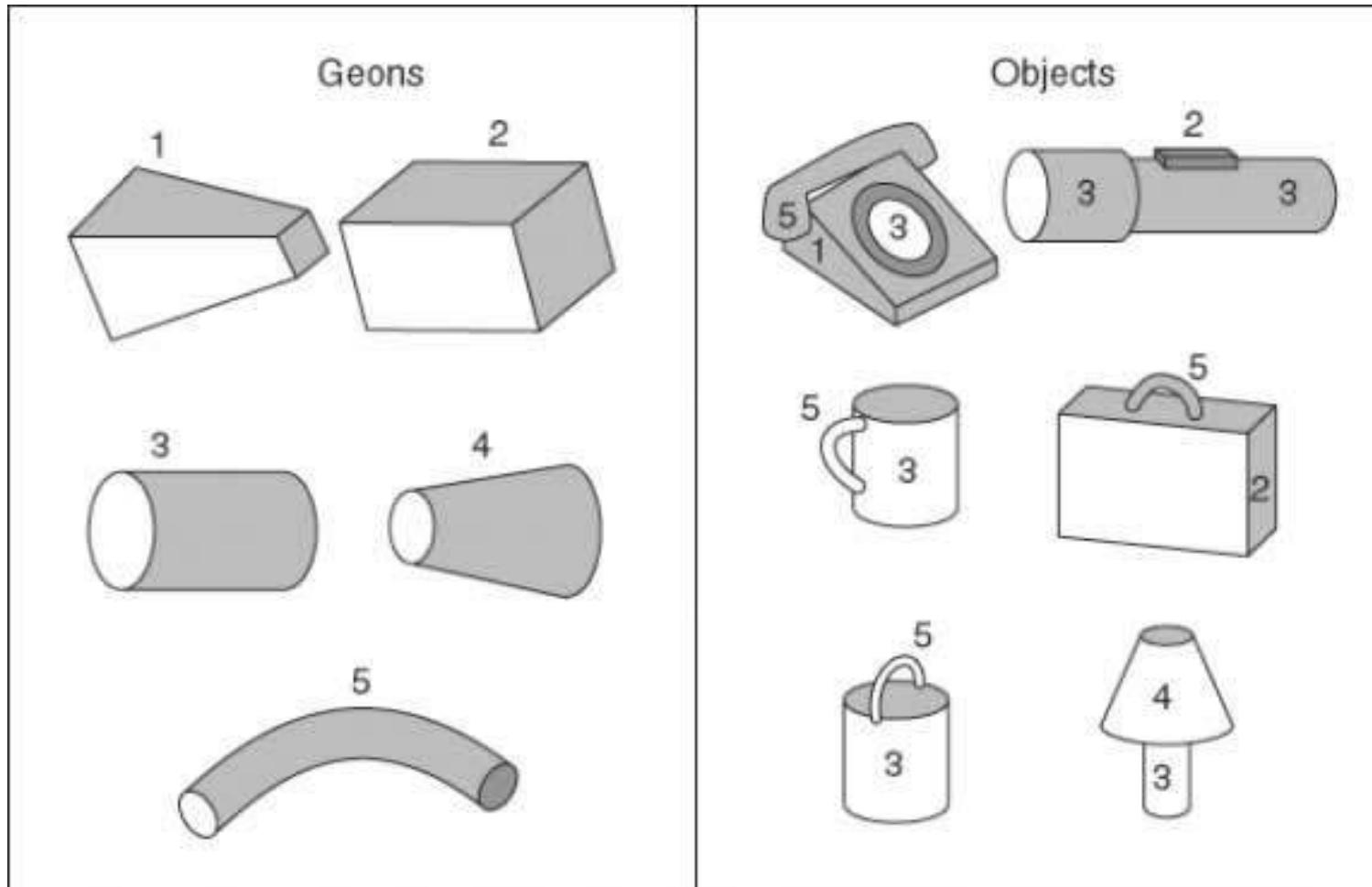
Hence, an object is an arrangement of a few simple three-dimensional shapes called *geometrical icons*, or **geons**.

Geons are simple volumes such as cubes, spheres, cylinders, and wedges.

Cube	Wedge	Pyramid	Cylinder	Barrel
				
Straight Edge Straight Axis Constant	Straight Edge Straight Axis Expanded	Straight Edge Straight Axis Expanded	Curved Edge Straight Axis Constant	Curved Edge Straight Axis Exp & Cont
Arch	Cone	Expanded Cylinder	Handle	Expanded Handle
				
Straight Edge Curved Axis Constant	Curved Edge Straight Axis Expanded	Curved Edge Straight Axis Expanded	Curved Edge Curved Axis Constant	Curved Edge Curved Axis Expanded

Object based: Recognition By Components

Different combinations of geons can be used to represent a large variety of objects

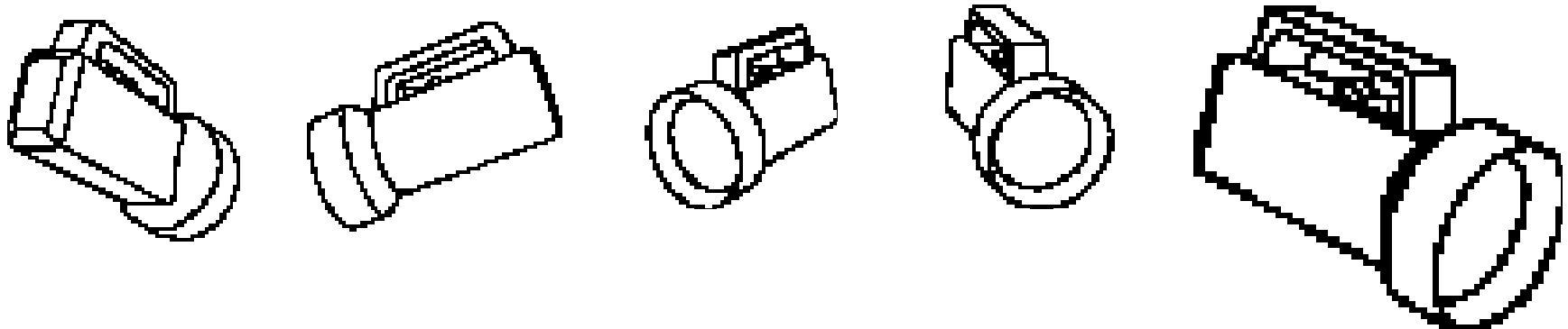


Object based: Recognition By Components

Geons are chosen to be:

- sufficiently different from each other to be easily discriminated
- robust to noise (can be identified even with parts of image missing)
- view-invariant (look similar from most viewpoints)

Different views of the same object are represented by the same set of geons, in the same arrangement. Therefore, the model achieves viewpoint invariance.

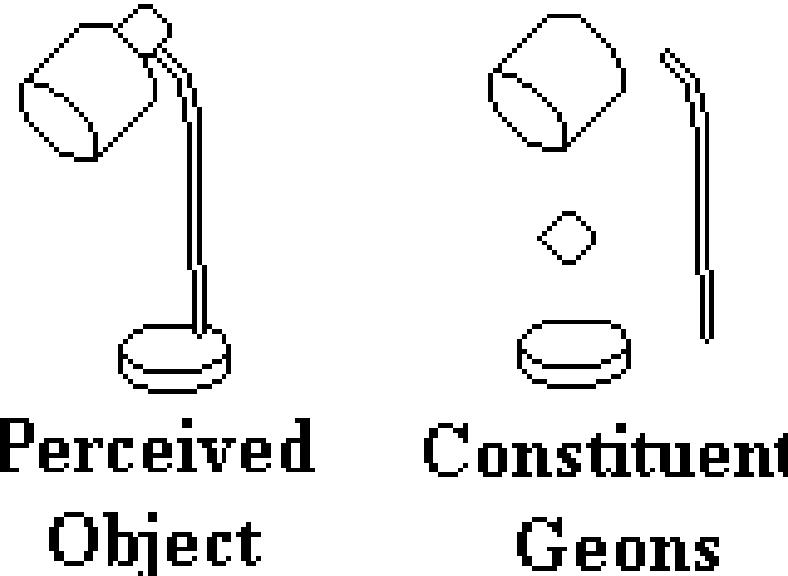


Object based: Recognition By Components

Matching

Recognition involves recognizing object elements (geons) and their configuration

The visual system parses an image of an object into its constituent geons.



Interrelations are determined, such as relative location and size (e.g., the lamp shade is left-of, below, and larger-than the fixture).

The geons and interrelations of the perceived object are matched against stored structural descriptions.

If a reasonably good match is found, then successful object recognition will occur.

Object based: Recognition By Components

Problems:

- difficult to decompose an image into components (i.e. to map an image onto a representation in geons)
- difficult to represent many natural objects using geons (may not have a simple parts-based description, e.g. a tree)
- cannot detect finer details which are necessary for identification of individuals or discrimination of similar objects. e.g.:

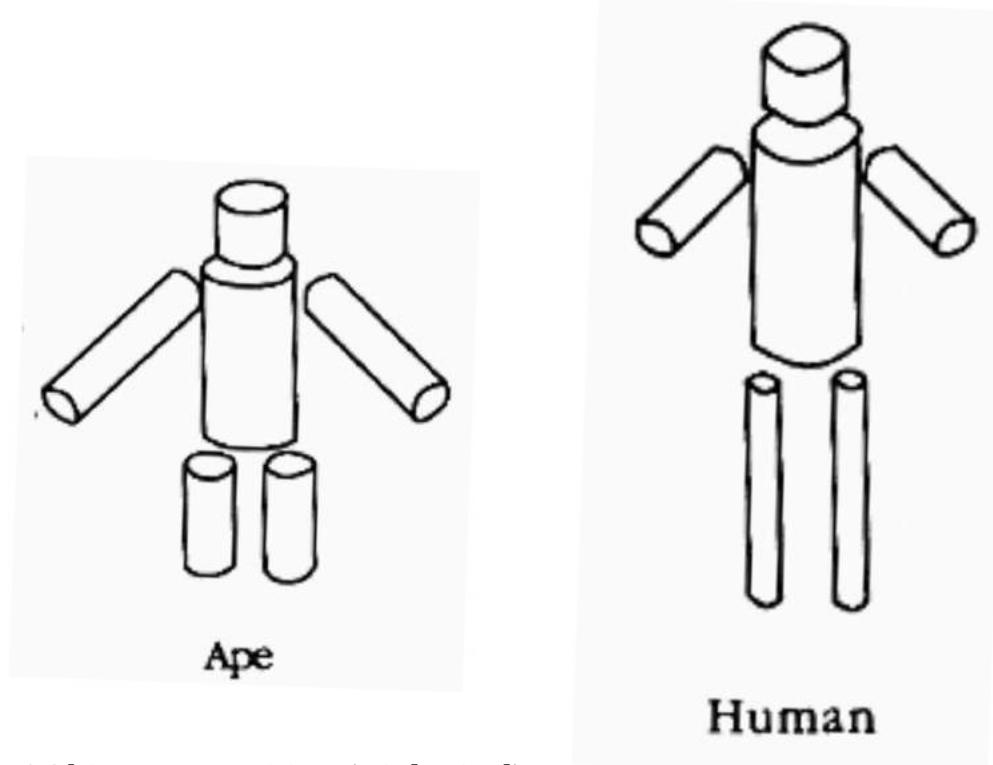


Image based

3D object **represented** by multiple, stored, 2D views of the object.

Object recognition occurs when a current pattern **matches** a stored pattern.

- Template matching
 - » An early version of the image-base approach.
 - » Too rigid to account for flexibility of human object recognition.
- Multiple Views approach
 - » More recent version of the image-based approach.
 - » Through experience, we encode multiple views of objects.
 - » These serve as the templates for recognition, but interpolation between stored views enables recognition of objects from novel viewpoints.

Configural vs Featural theories



Who is this?

Is he looking well?

Configural vs Featural theories



Configural vs Featural theories



Inverted faces: featural processing

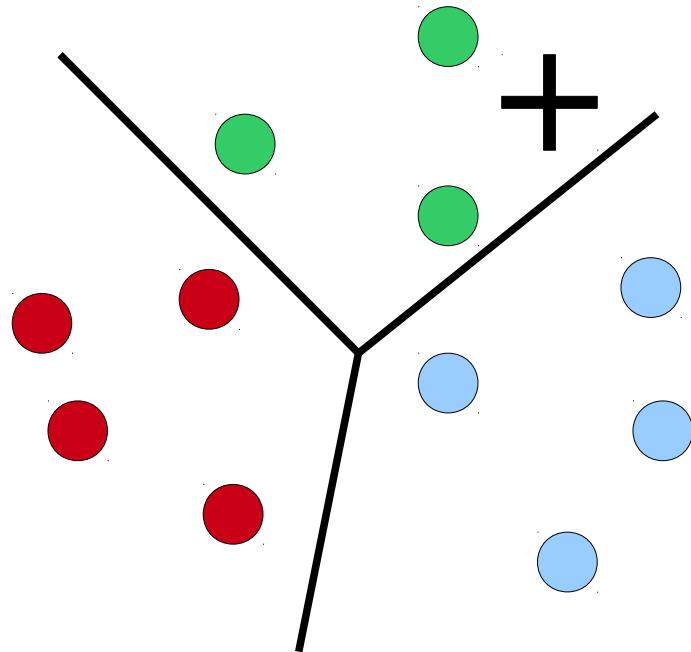
- features processed independently, relationships between features ignored.



Upright faces: configural processing

- holistic, global

Rules vs Prototypes vs Exemplars



How are the boundaries between different categories defined?

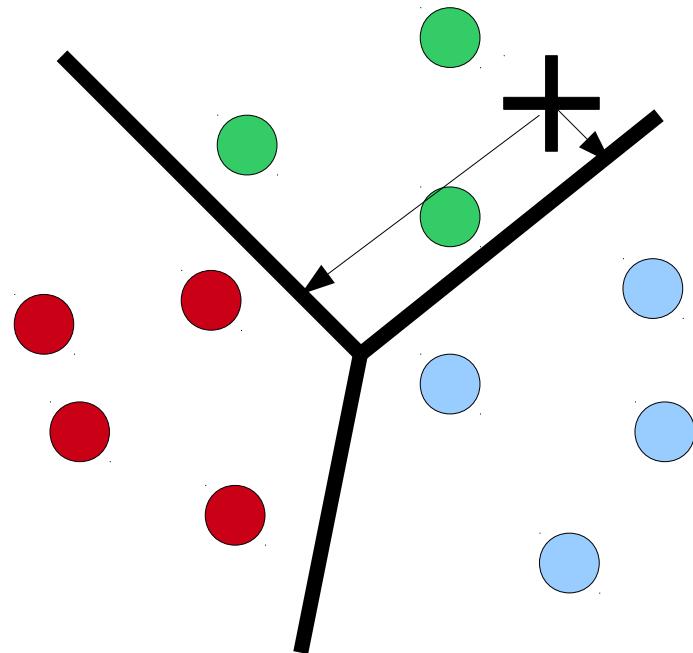
How are new stimuli assigned to the closest category?

feature space (2D in this example) – see lecture 5

● ● ● = previous examples of stimuli from 3 different categories

⊕ = a new stimulus from an unknown category

Rules



Category membership defined by abstract rules, e.g.

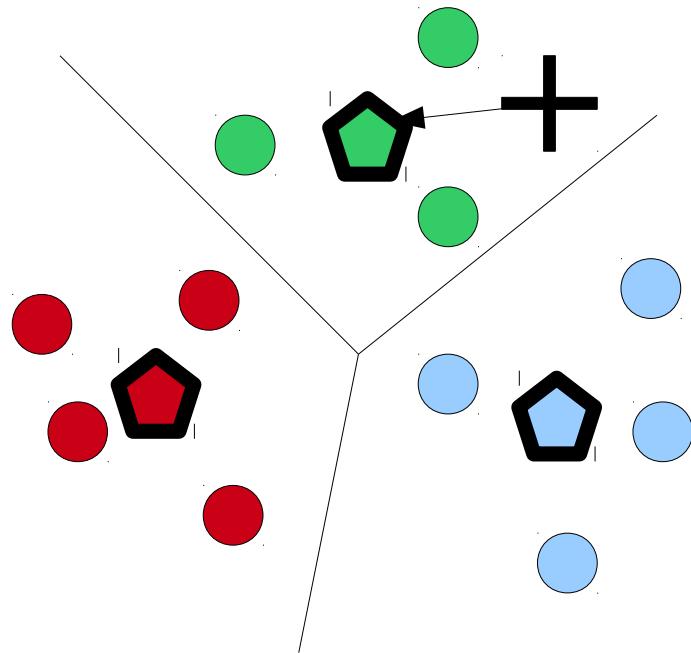
- has three sides = triangle
- has four legs and barks = dog
- has a beak and feathers = bird

Anything that satisfies the rule(s) for the category goes into that category

For: over-extension of rules of grammar, e.g. “goed” instead of went, “bitted” instead of bitten, “mouses” instead of mice.

Against: Some members are better examples of a category (graded membership), e.g. bear is a better mammal than a whale, 4 is a better even number than 106, pigeon is a better bird than penguin.

Prototypes



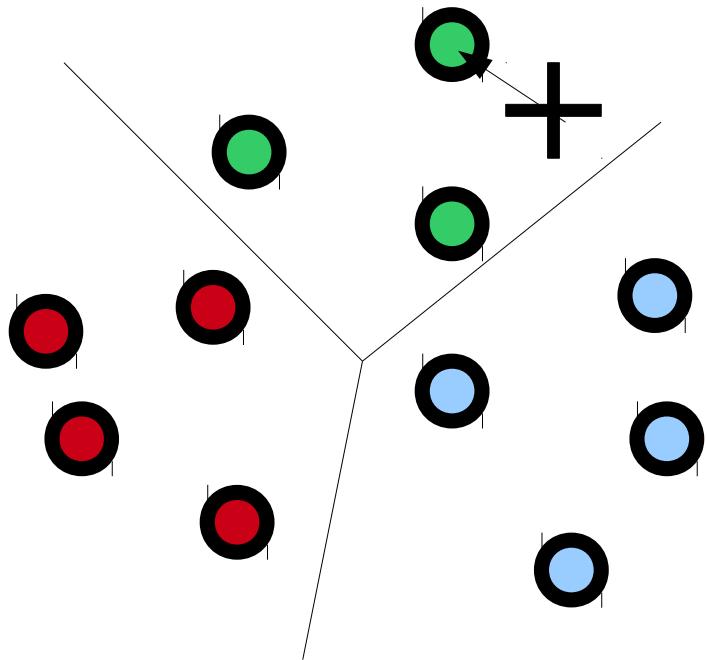
Calculate the average (or prototype) of all the individual instances from each category.

A new stimulus is compared to the stored prototypes and assigned to the category of the nearest one

For: prototypical category members are accessed more quickly and learnt more easily (e.g. pigeons vs penguins.)

Against: variations within a class can not be represented.

Exemplars



Specific individual instances of each category (“exemplars”) stored in memory.

A new stimulus is compared to the stored exemplars and assigned to the category of the nearest one

For: successfully predicts some kinds of mis-categorizations (e.g., a whale as a fish).

Against: Some members are better examples of a category (graded membership), e.g. bear is a better mammal than a whale, 4 is a better even number than 106, pigeon is a better bird than penguin.

Classifiers

Prototype and Exemplar theories in psychology correspond to standard **classification** methods used in pattern recognition / machine learning.

These methods use “supervised” learning:

- assumes that class for each data point in the training set is known
- new (unknown) data points assigned to appropriate class based on similarity to training examples

The alternative is unsupervised learning:

- assumes that class for each data point is unknown
- all data points assigned to appropriate class based on similarity

We previously came across unsupervised pattern recognition / machine learning methods, called **clustering**, when discussing image segmentation techniques (i.e. k-means clustering, agglomerative hierarchical clustering, graph cutting).

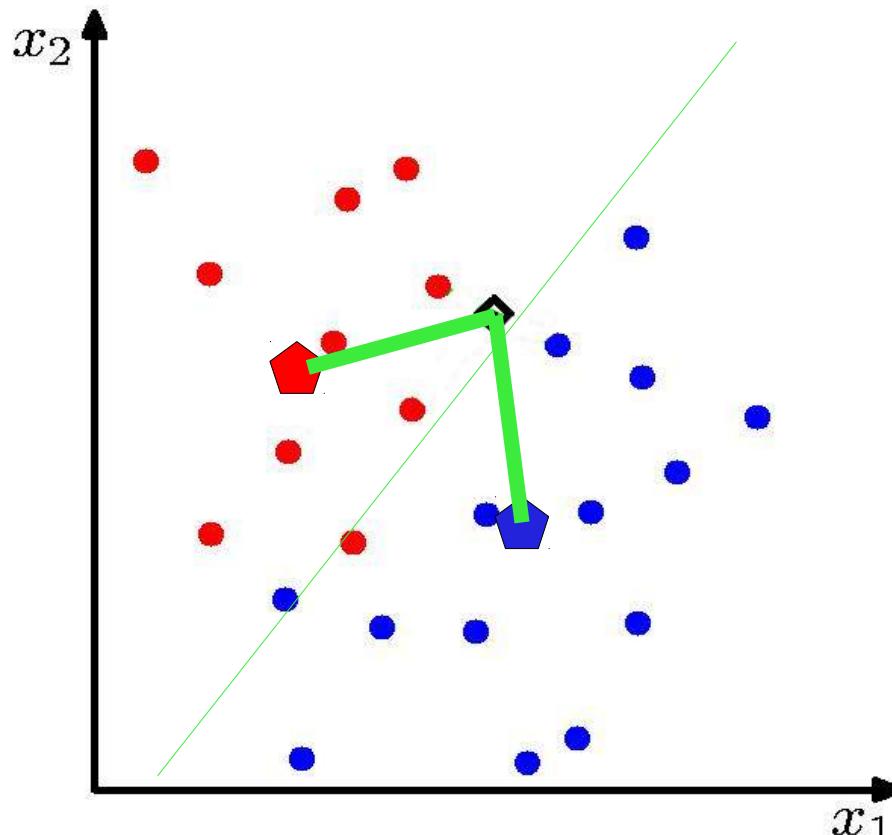
Nearest Mean Classifier (Prototype)

For each class

- calculate the mean of the feature vectors for all the training examples in that class

For each new stimulus

- find the closest class prototype and assign new stimulus to that class label



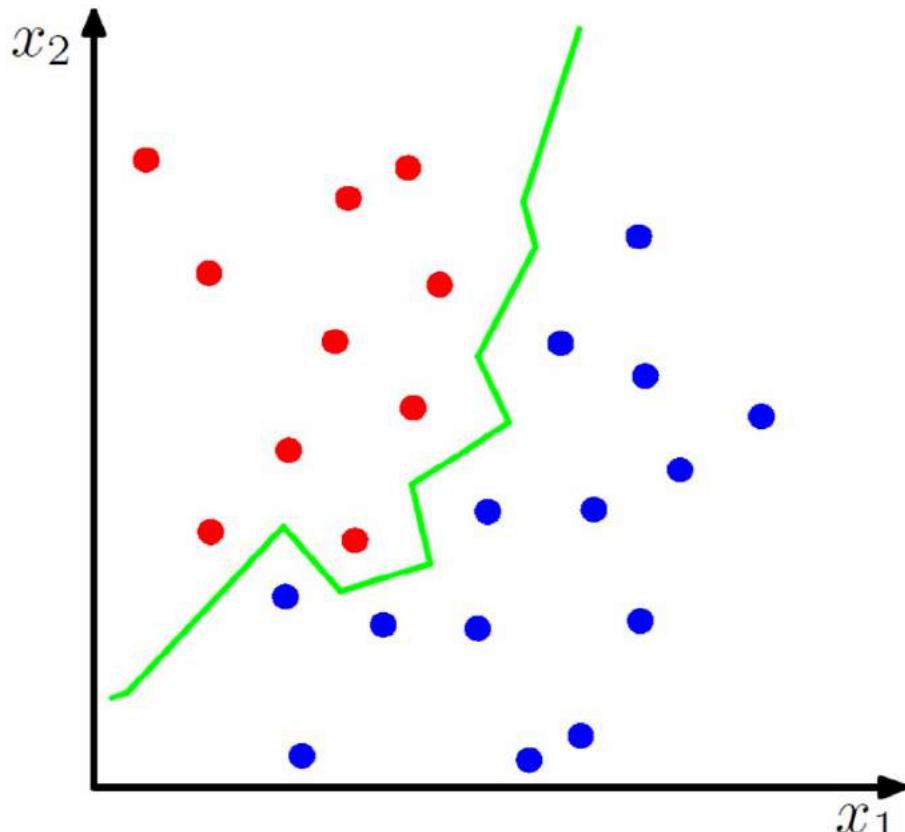
Decision boundary is linear. Hence, suitable only if data is linearly separable

Nearest Neighbour Classifier (Exemplar)

- Save the vectors for all the training examples (instead of just the mean for each class)

For each new stimulus

- find the closest training exemplar and assign new stimulus to that class label



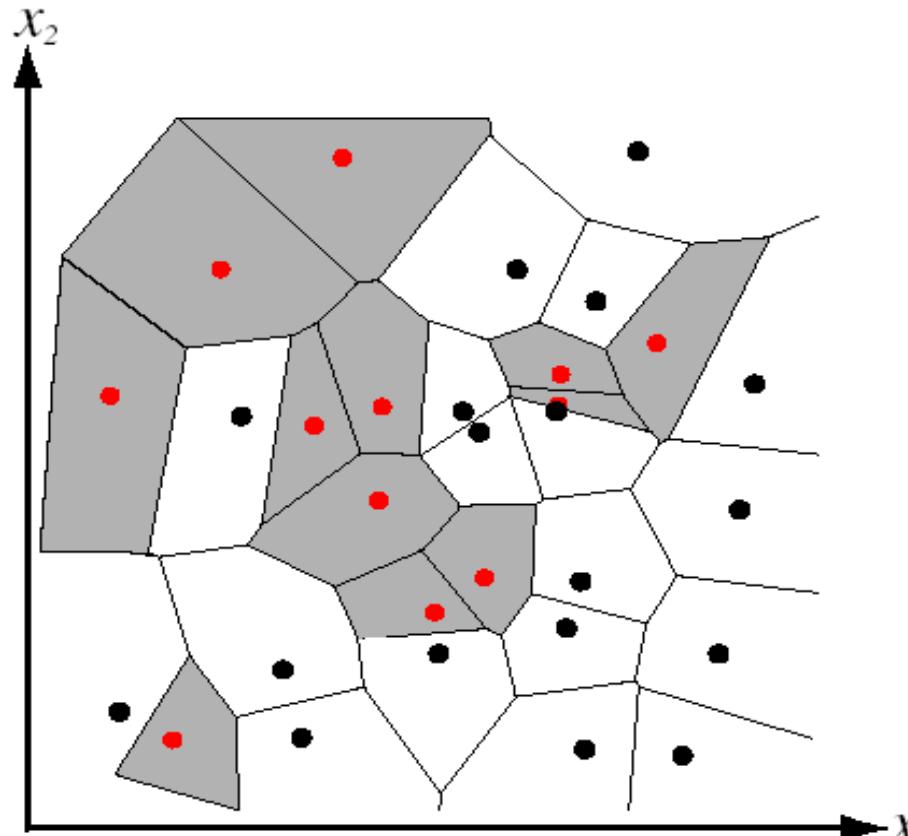
Decision boundary is non-linear (piecewise linear). Hence, suitable if data is non-linearly separable.

Nearest Neighbour Classifier (Exemplar)

- Save the vectors for all the training examples (instead of just the mean for each class)

For each new stimulus

- find the closest training exemplar and assign new stimulus to that class label



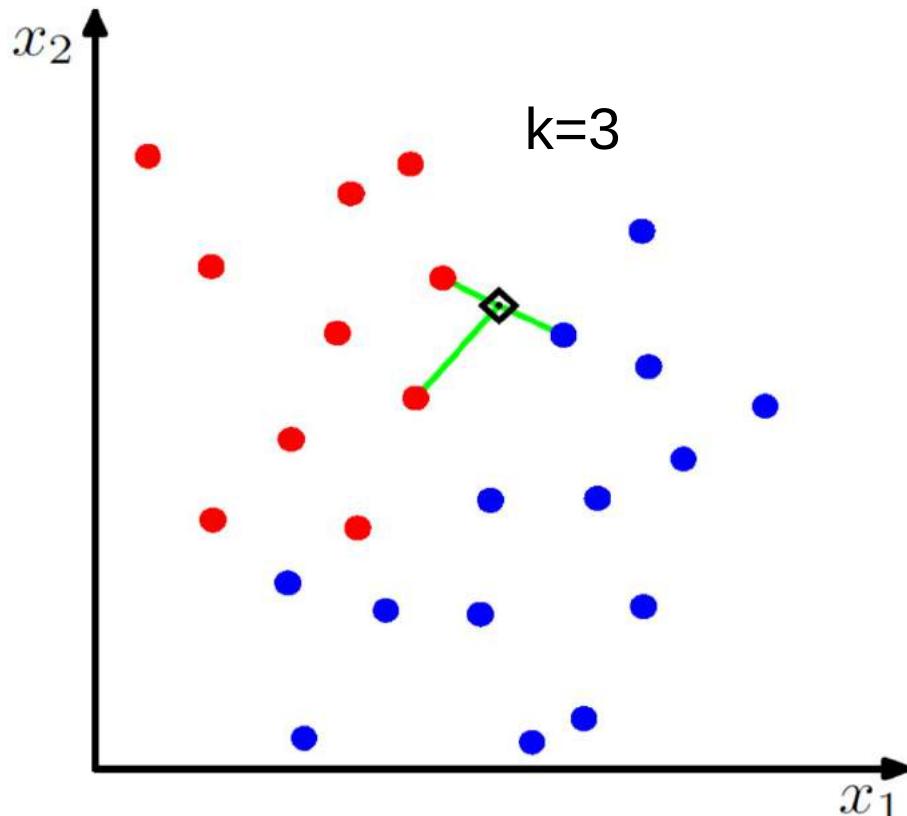
Decision boundaries form
Voronoi partitioning of
feature space.
Doesn't deal with outliers.

K-Nearest Neighbours Classifier

- Save the vectors for all the training examples (instead of just the mean for each class)

For each new stimulus

- find the k closest training exemplars and assign new stimulus to the class label of the majority of these points (e.g. closest points vote on correct label)



Decision boundary is non-linear. Hence, suitable if data is non-linearly separable.

k typically small and odd (to break ties).

Increasing k reduces the effects of outliers

Similarity Measures

Determining the nearest neighbour(s) or nearest mean requires some measure of the distance between two sets of features.

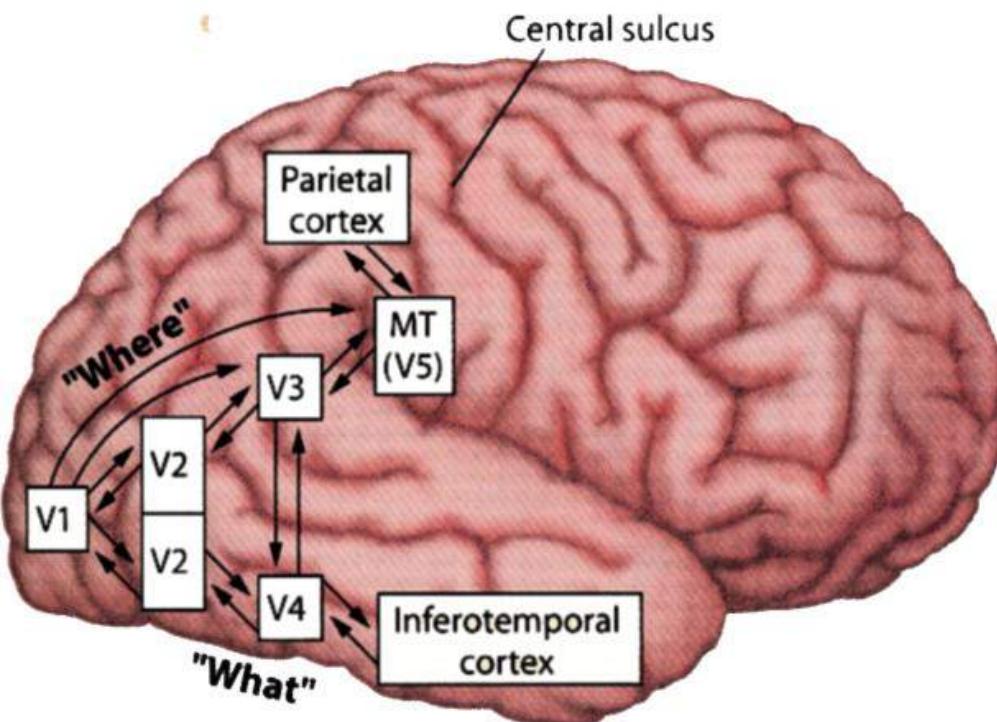
As previously, we can either find the minimum distance, e.g.:

- Sum of Squared Differences (SSD)
- Euclidean distance
- Sum of Absolute Differences (SAD) = Manhattan distance

Or, find the maximum similarity, e.g.:

- Cross-correlation
- Normalised cross-correlation
- Correlation coefficient

The Cortical Visual System: pathways



“What” and “Where” pathways

Hierarchically organised:

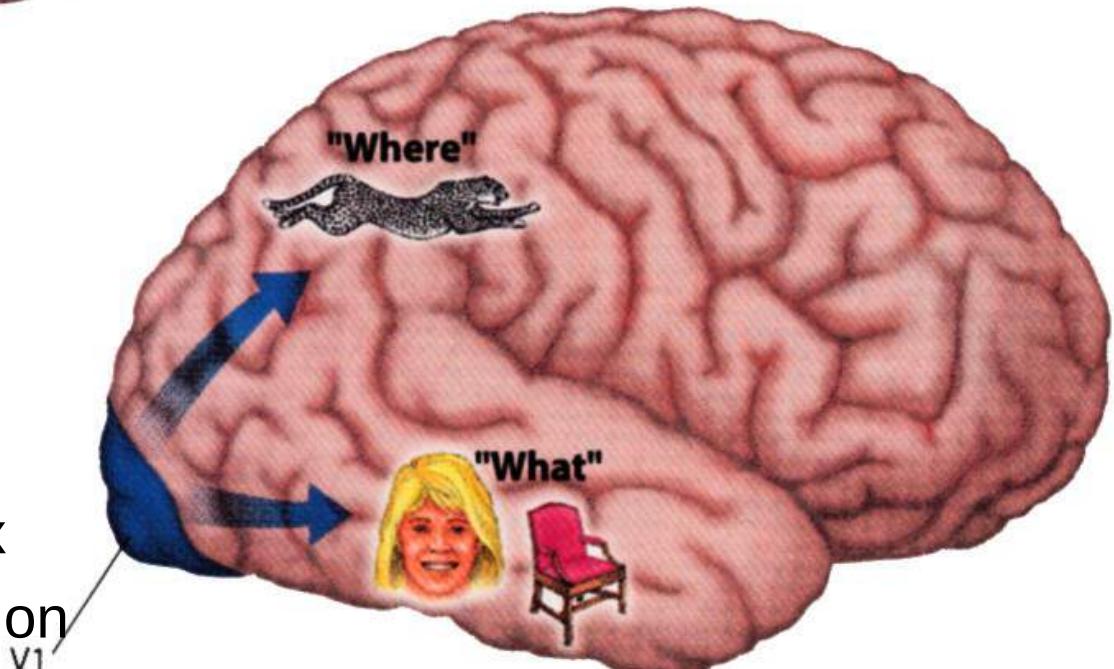
- simple, local, RFs at V1
- complex, large, RFs in higher areas

Where (or How):

- V1 to parietal cortex
- spatial / motion information

What

- V1 to inferotemporal cortex
- identity / category information

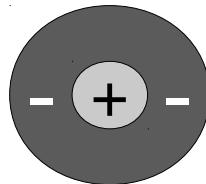


Hierarchy of Receptive Fields

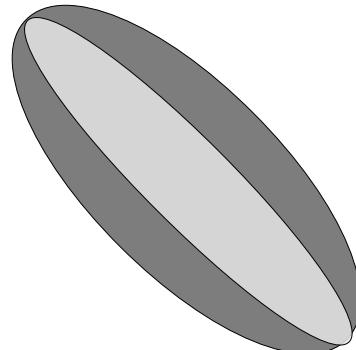
As we progress along a pathway, neurons' preferred stimuli gets more complex, receptive fields become larger, and there is greater invariance to location.

e.g: Eye → LGN → V1

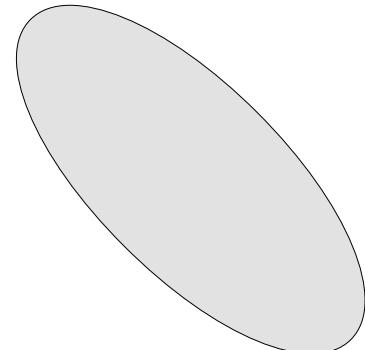
Centre-surround Cells → Simple Cells → Complex Cells



Centre-surround cells respond to isolated spots of contrasting intensity or colour



Simple cells respond to edges/bars at a specific orientation with specific contrast polarity at a specific location



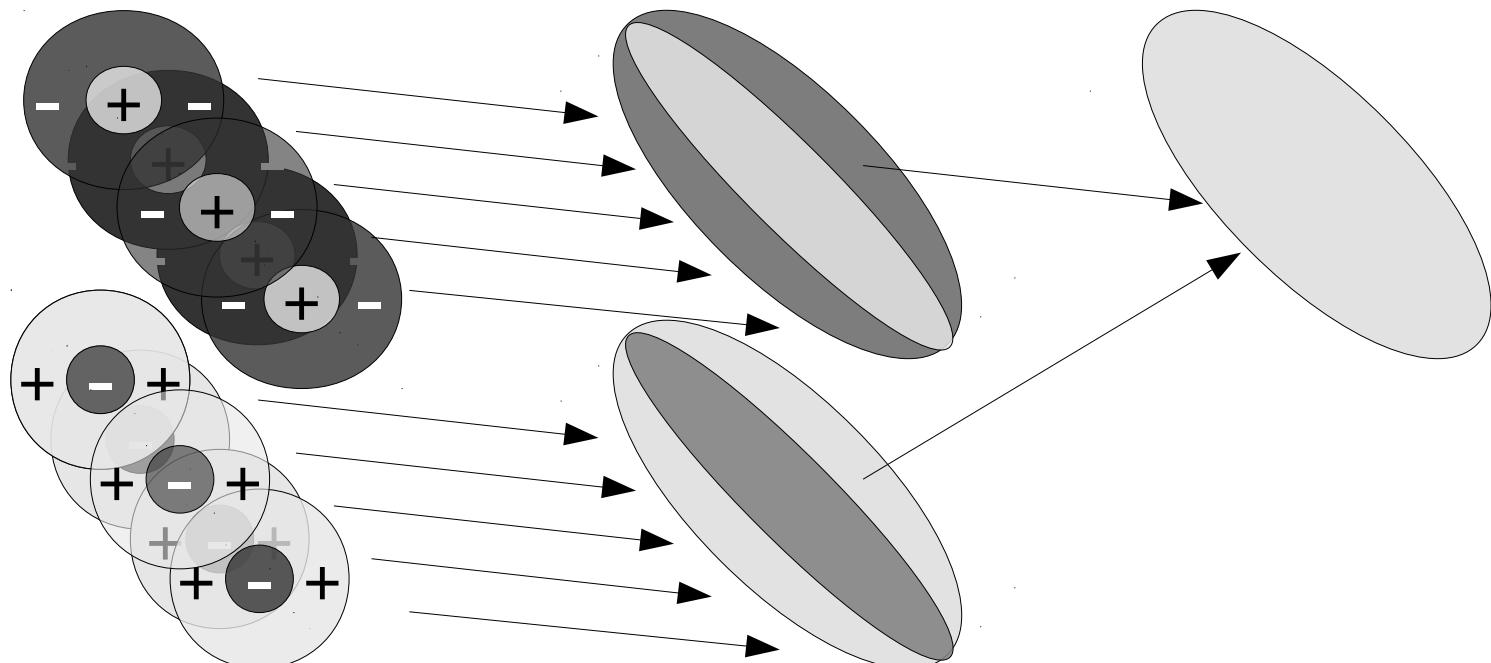
Complex cells respond to edges/bars of a specific orientation with any polarity at any position within a small region

Hierarchy of Receptive Fields

More complex RFs built by combining outputs from multiple cells with simpler RFs.

e.g: Eye → LGN → V1

Centre-surround Cells → Simple Cells → Complex Cells



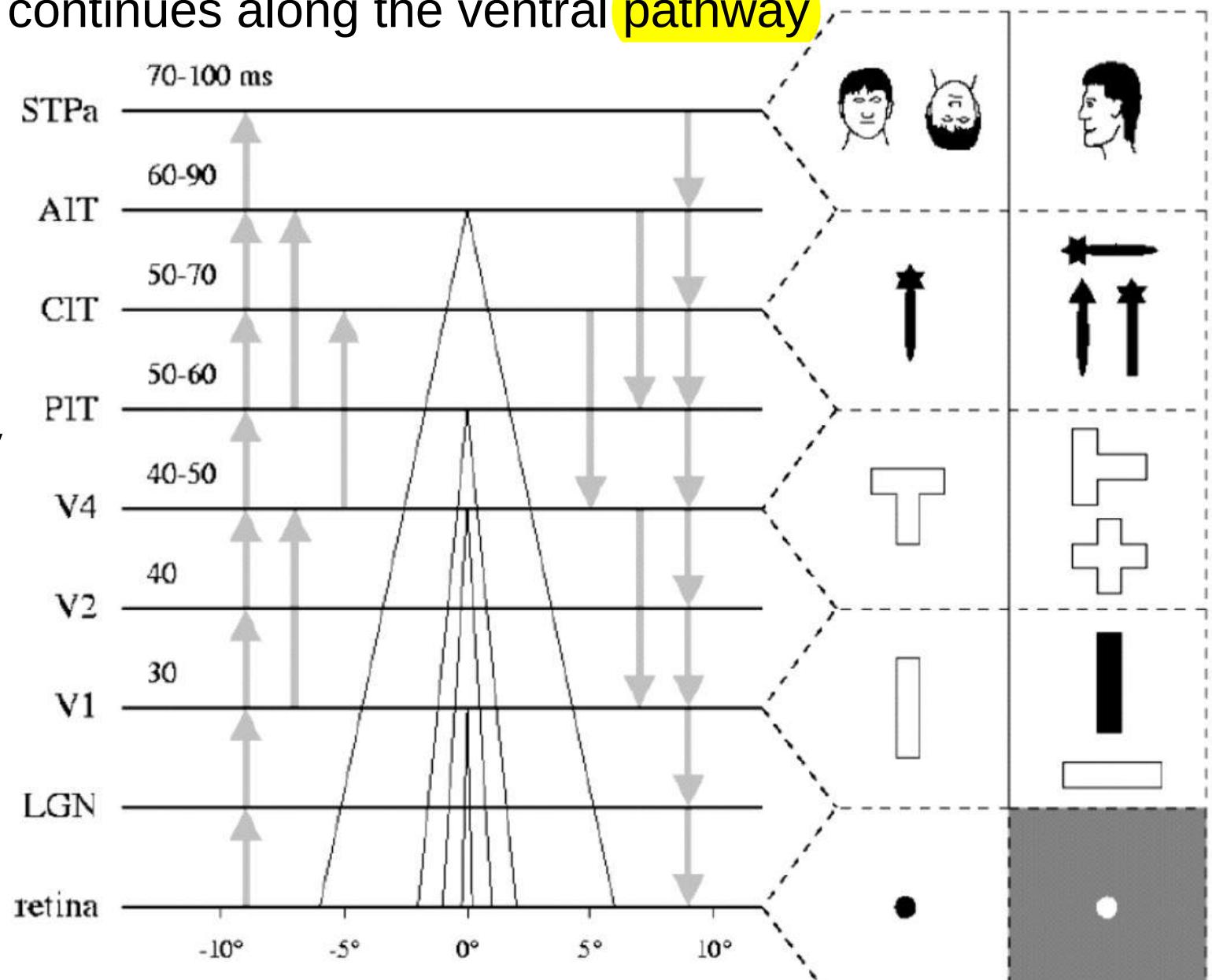
Simple cells respond
when multiple co-
aligned centre-surround
cells are active

Complex cells respond
when any of multiple
similarly oriented
simple cells are active

Hierarchy of Receptive Fields

This trend continues along the ventral pathway

- larger RFs
- higher complexity
- higher invariance



Hierarchy of Receptive Fields

Neurons' preferred stimuli gets more complex but they have less sensitivity to location.

Neurons near the end of the ventral pathway respond to very complex stimuli, like faces.



V2	V4	posterior IT	anterior IT

Feedforward models of cortical hierarchy

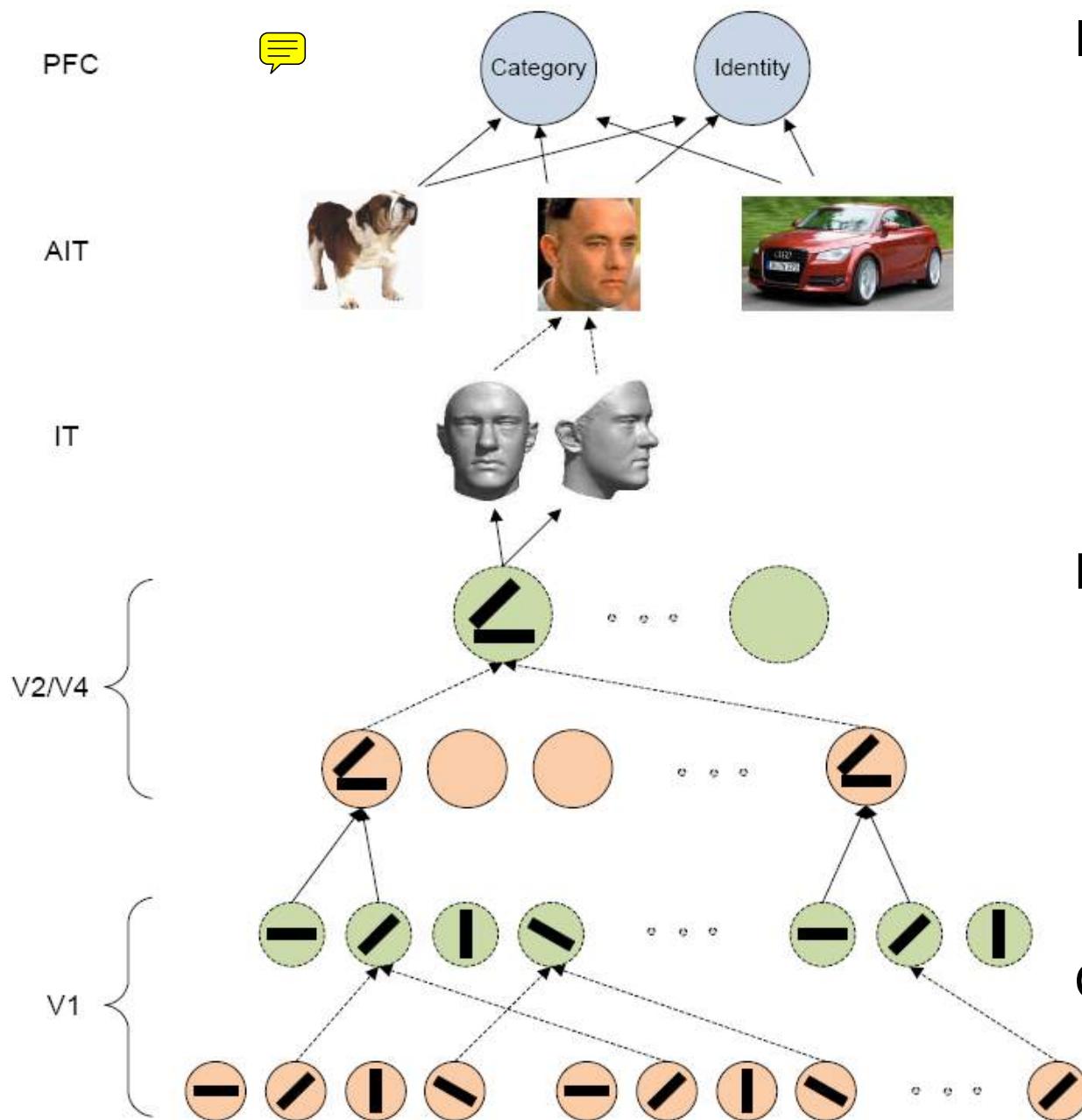
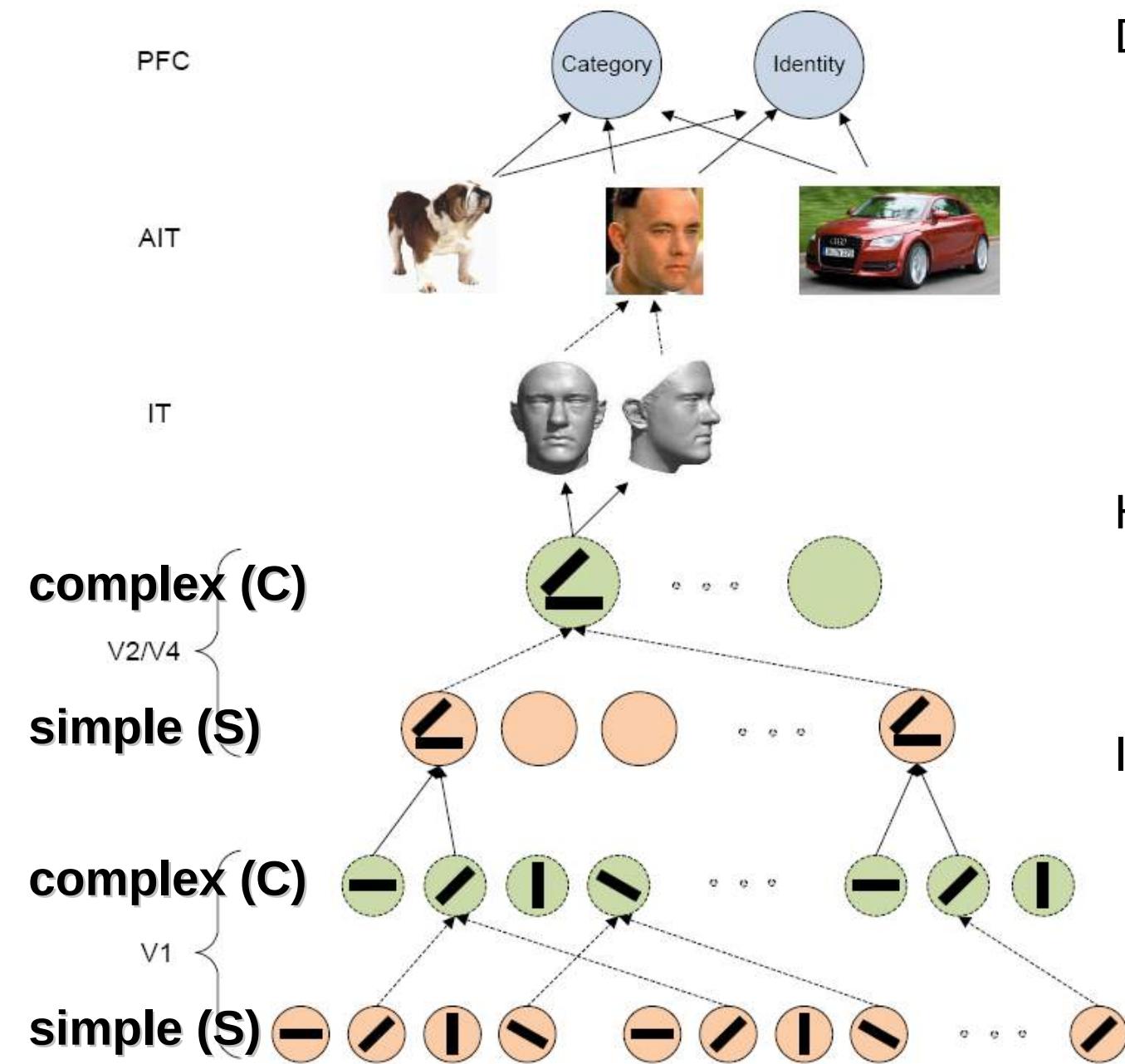


Image processed by layers of neurons with progressively more complex receptive fields at progressively less specific locations.

Hierarchical in that features at one stage are built from features at earlier stages.

Can be thought of as hierarchical template matching

Feedforward models: HMAX



Different mathematical operations are required to increase the complexity or selectivity of RFs and to increase the degree of invariance of RFs.

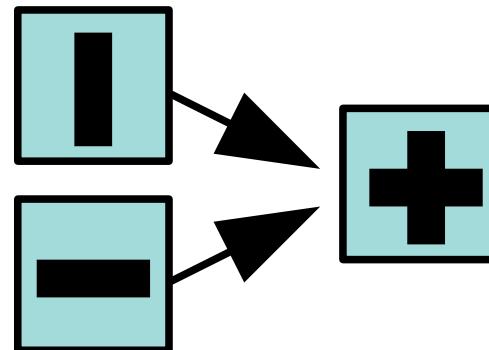
Hence, several models use alternating layers of neurons with different properties.

In analogy with the response properties of V1, these are called **simple (S)** and **complex (C)** cells.

Feedforward models: HMAX

Unit types

Simple
“S-cells”



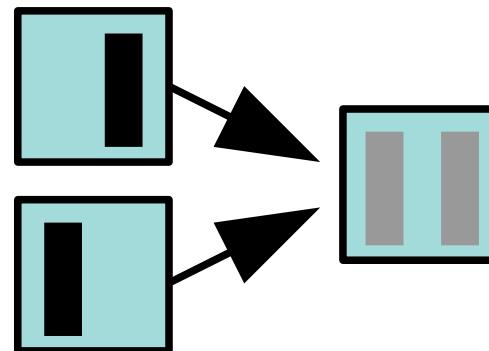
Computation

sum
“and”-like

Result

Increased
Selectivity

Complex
“C-cells”

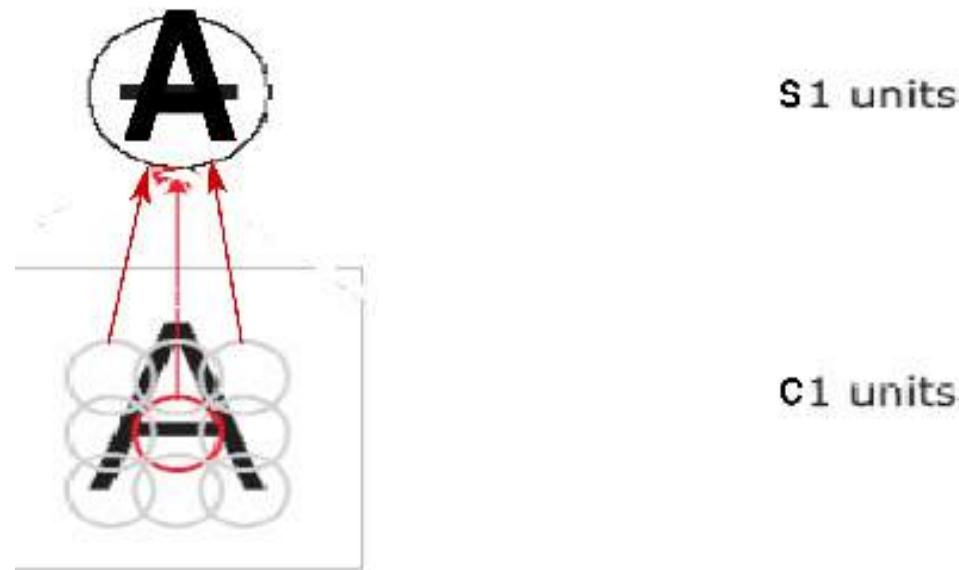


max
“or”-like

Increased
Invariance

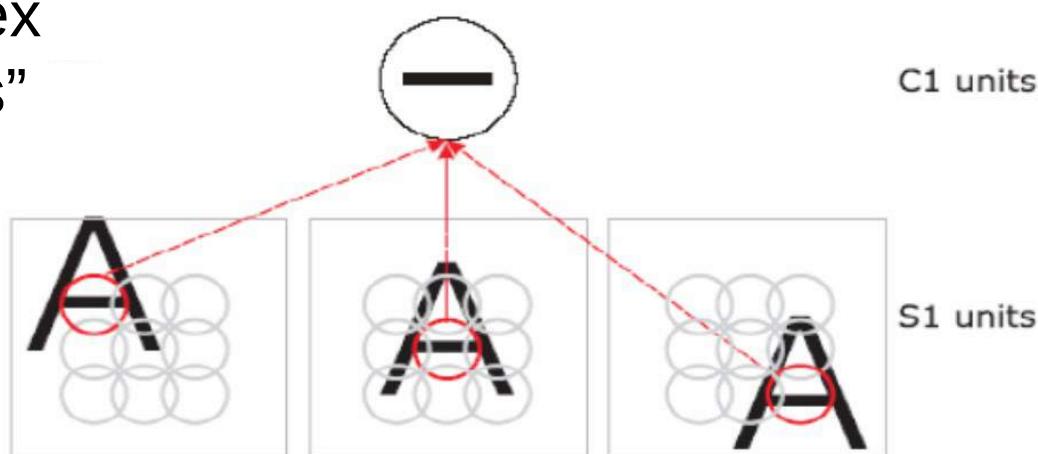
Feedforward models: HMAX

Simple
“S-cells”



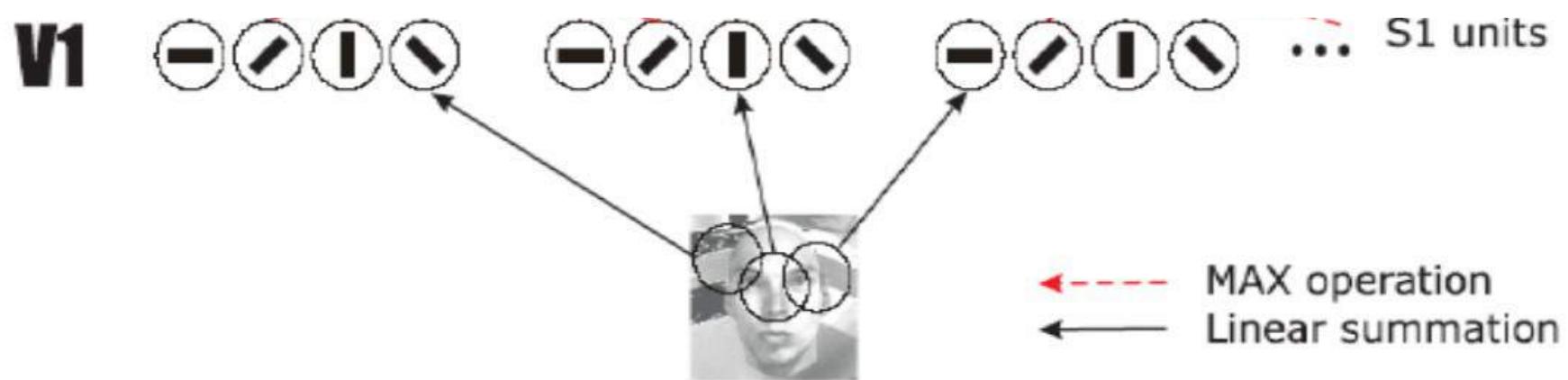
S-cells in one layer respond to conjunctions of C-cells in previous layer.

Complex
“C-cells”

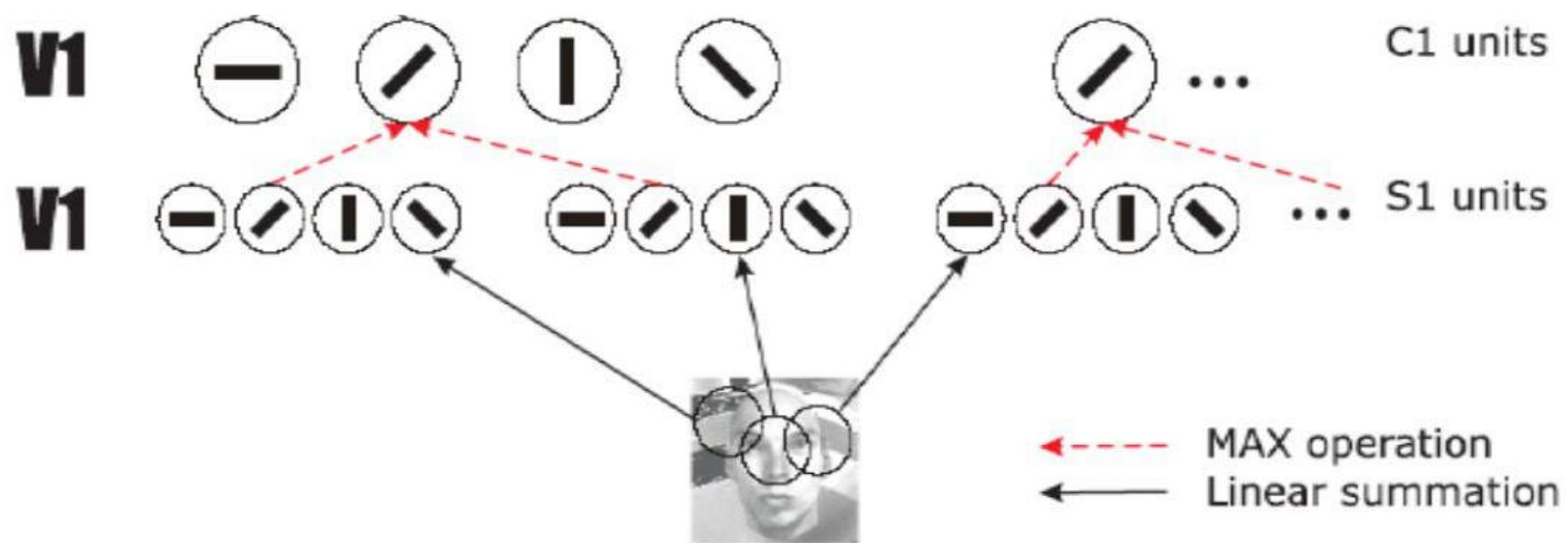


C-cells in one layer respond to any S-cell in a small neighborhood of the previous layer.

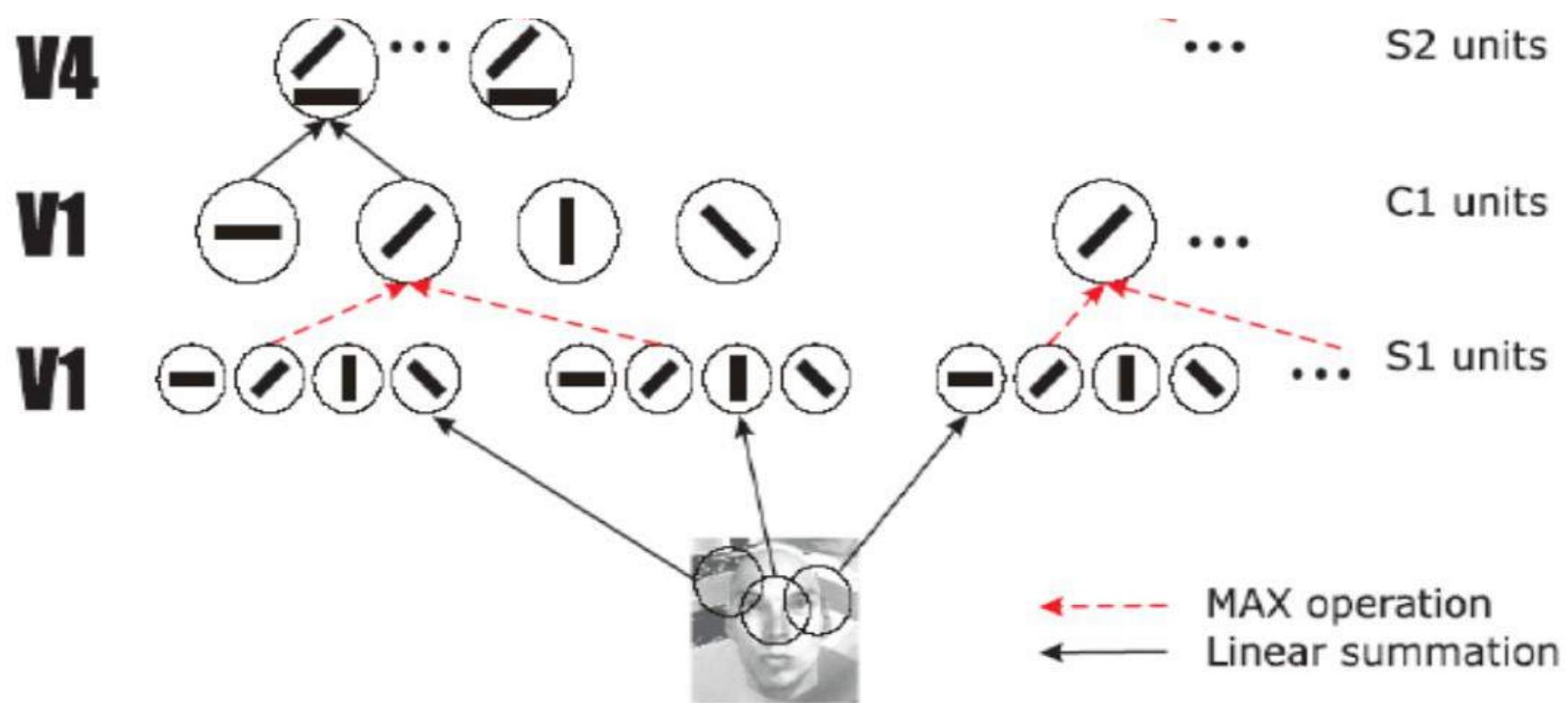
Feedforward models: HMAX



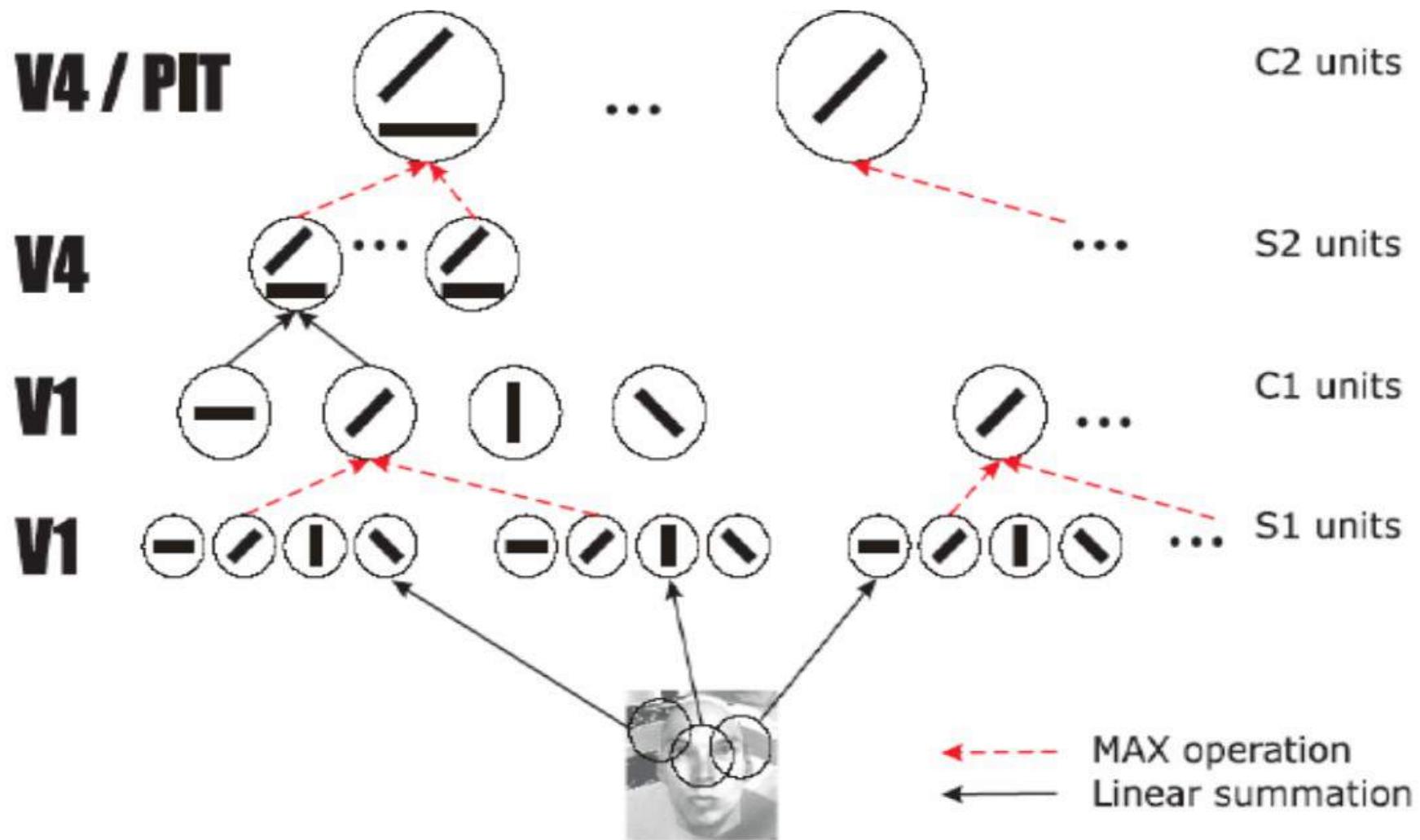
Feedforward models: HMAX



Feedforward models: HMAX

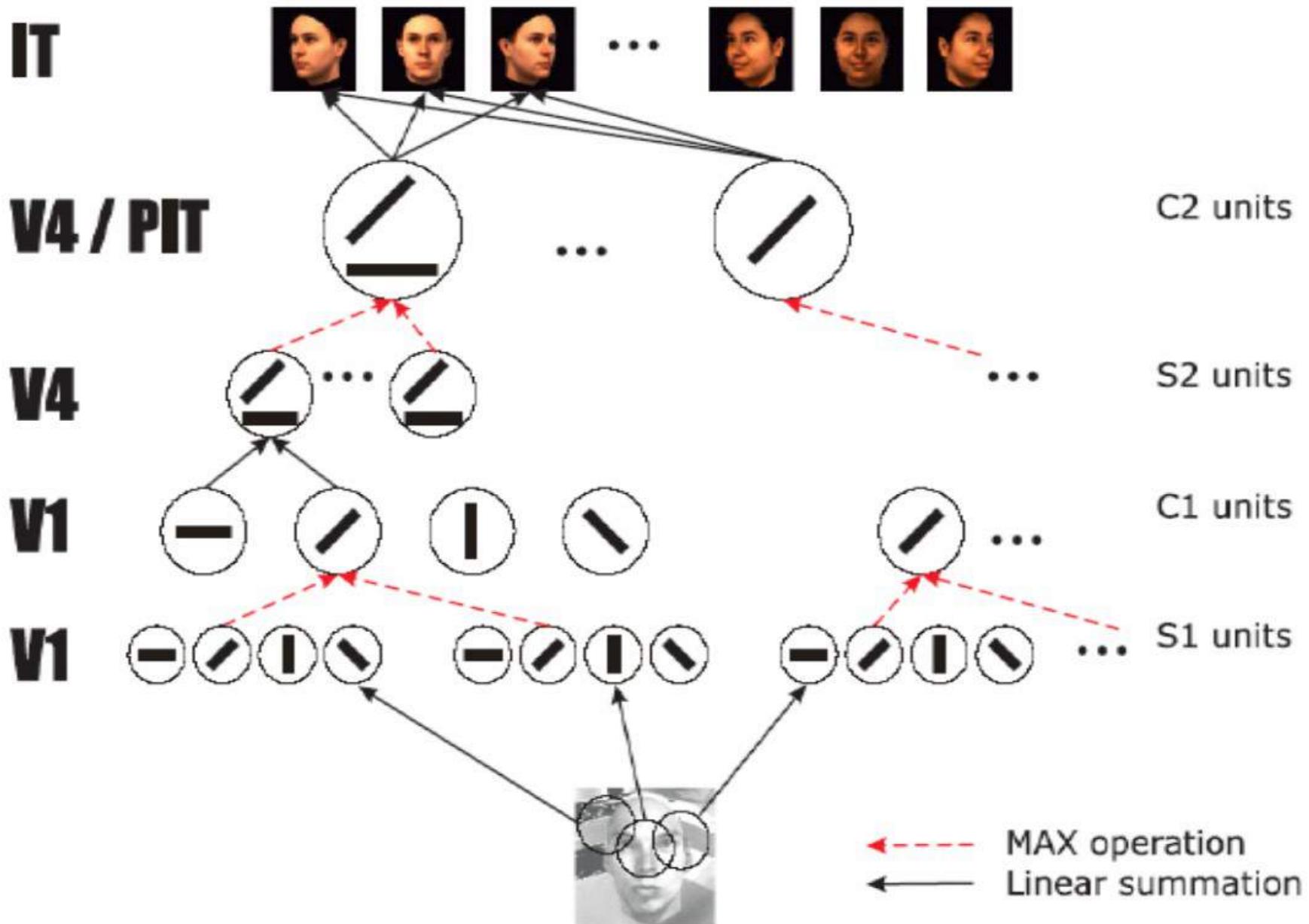


Feedforward models: HMAX

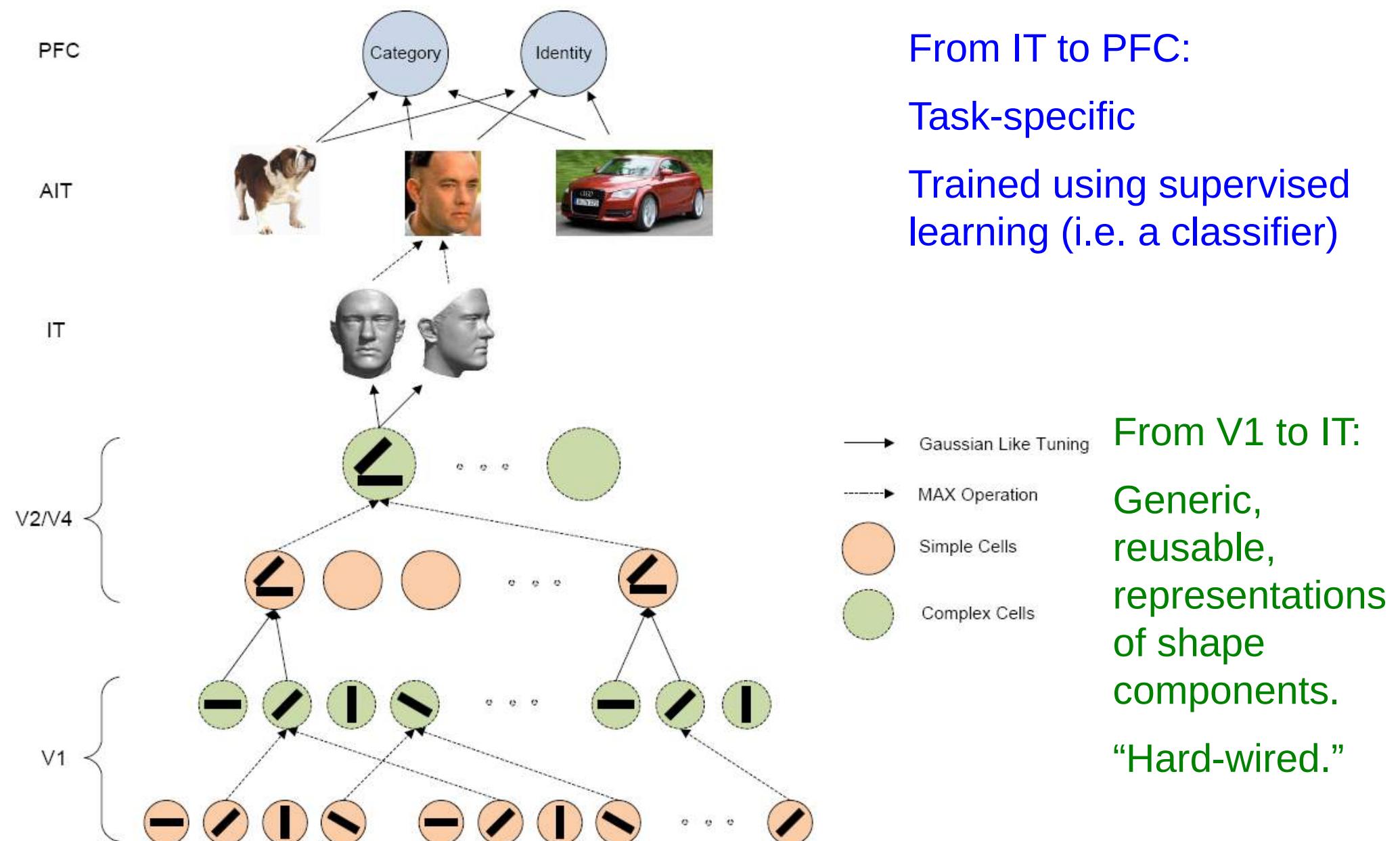


Feedforward models: HMAX

View-tuned units (VTUs). Ex: race units

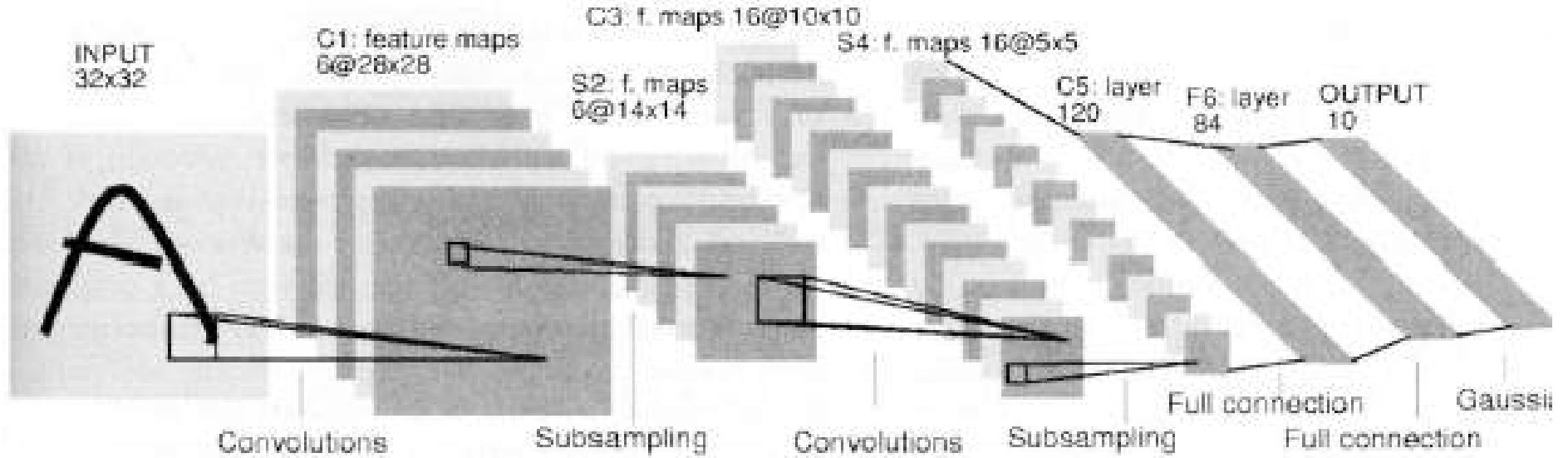


Feedforward models: HMAX



Feedforward models: CNN

CNN = convolutional neural network

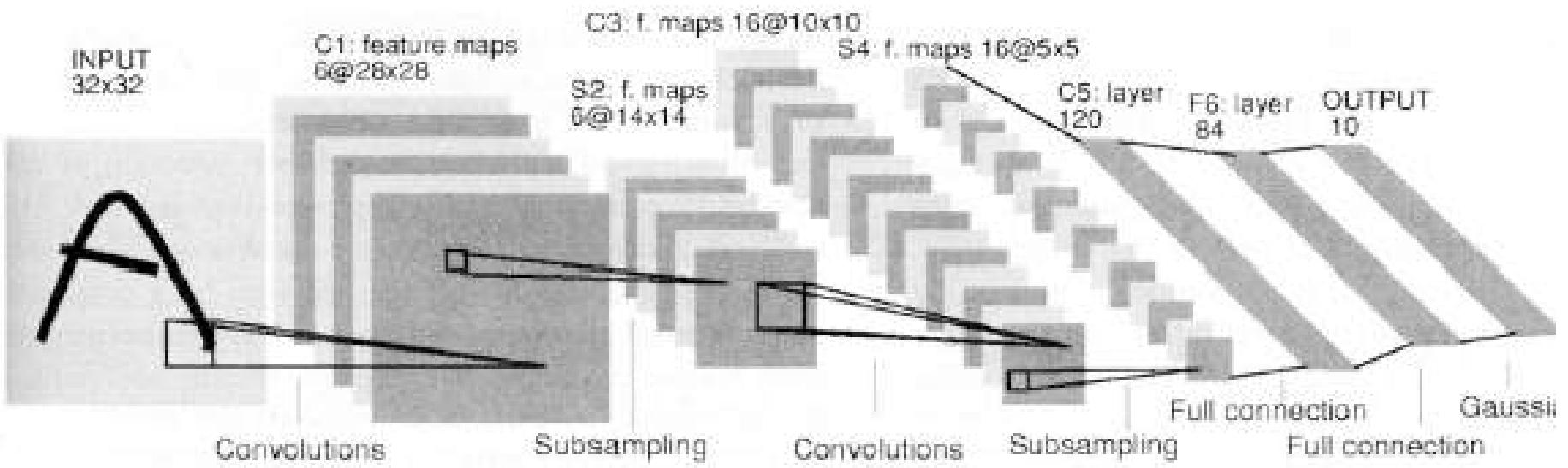


A hierarchical model similar to HMAX can be implemented using standard image processing techniques: convolution and sub-sampling.

It consists of alternating layers of

- convolution (equivalent to responding to conjunctions), and
- sub-sampling (equivalent to responding to any input in a small neighbourhood, to reduce location specificity).

Feedforward models: CNN

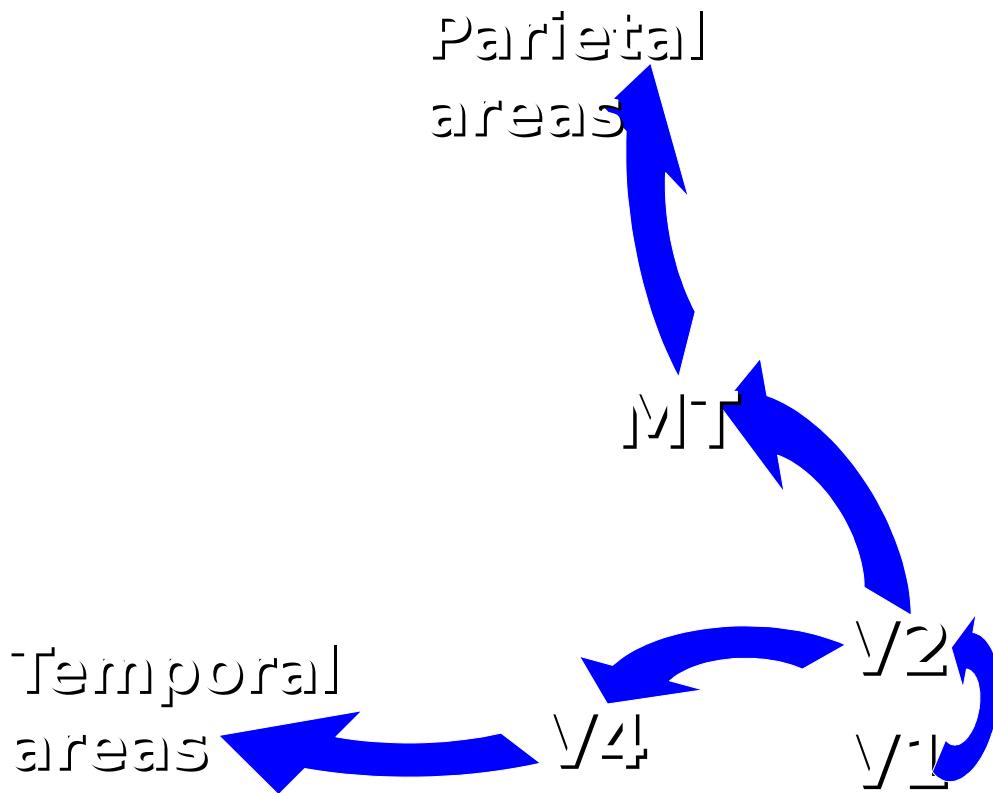


Confusingly:

- convolution layers are called “C layers” but are equivalent to S layers in HMAX, and
- sub-sampling layers are called “S layers” but are equivalent to C layers in HMAX.

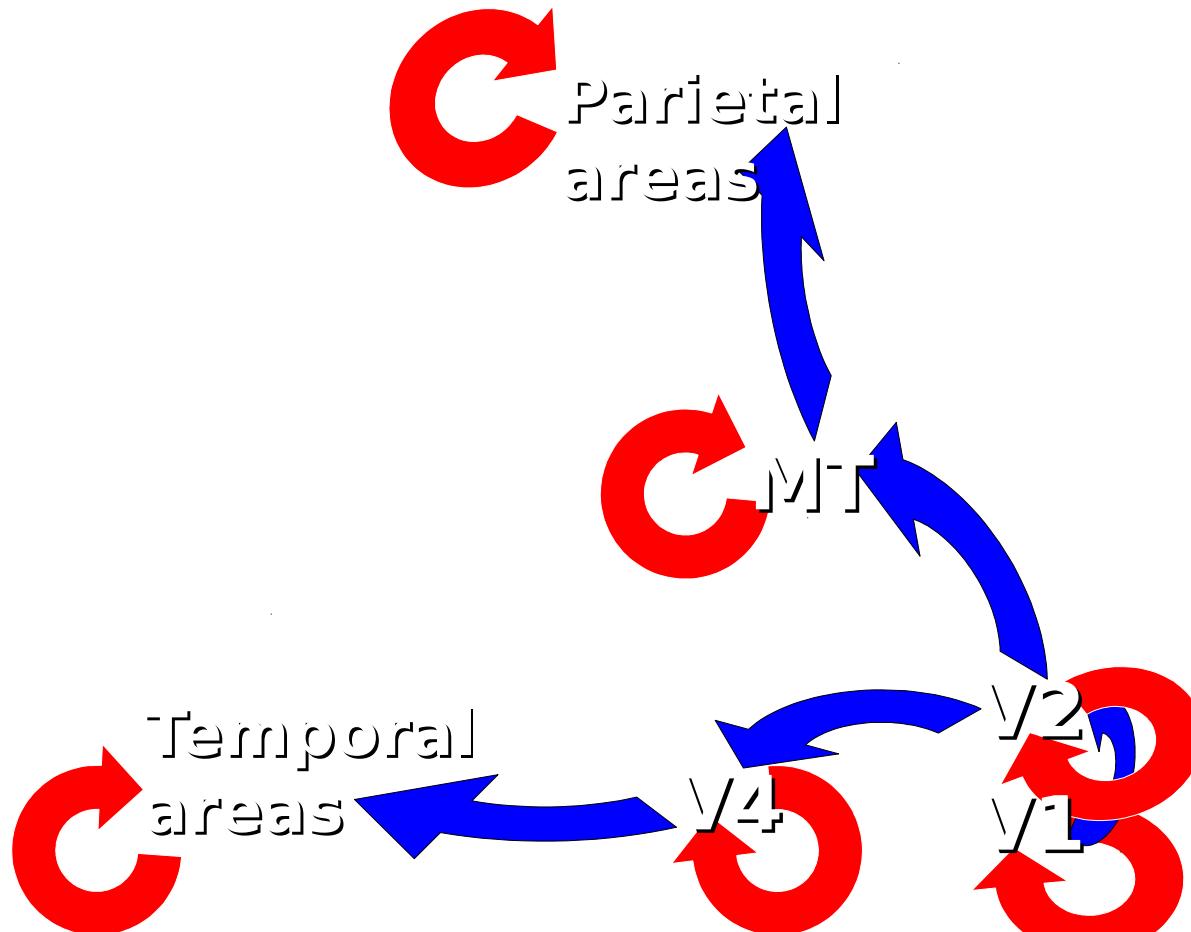
Deep Neural Networks (like HMAX and CNN) produce state-of-the-art performance in many computer vision tasks.

Feedforward models of cortical hierarchy



HMAX and CNN are two examples of several models that propose a purely serial, feedforward, sequence of cortical information processing.

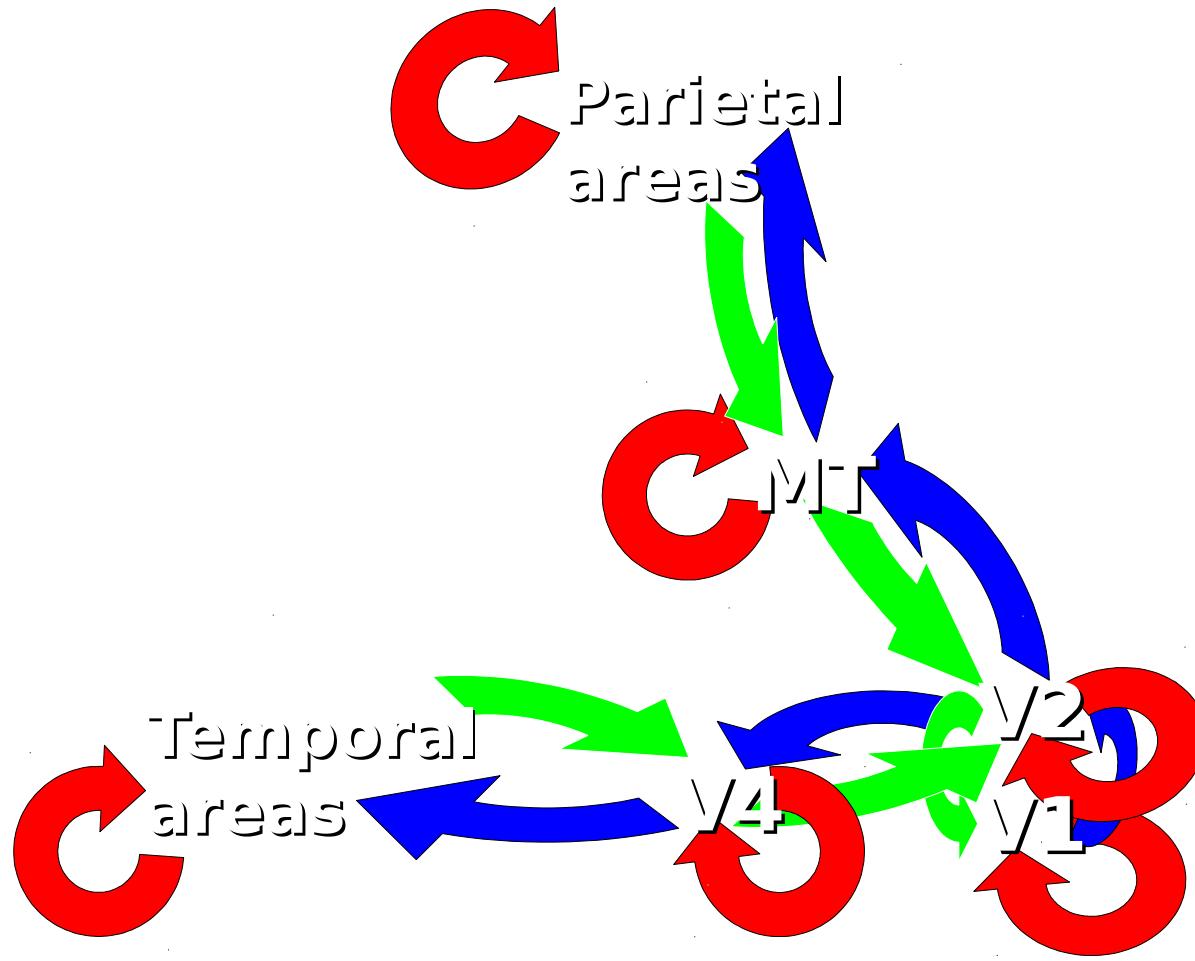
Recurrent models of cortical hierarchy



However, there are two types of recurrent connections.

(1) Within each region, lateral connections (both excitatory and inhibitory) enable neurons within the same population to interact (see descriptions of V1 and V2 in earlier lectures).

Recurrent models of cortical hierarchy



In addition,

(2) feedback connections convey information from higher cortical regions to primary sensory areas.

Bottom-up and top-down information interacts to affect perception.

Recurrent models of cortical hierarchy

Allow bottom-up and top-down information to be combined.

- **Bottom-Up processes:**
 - Using the information in the stimulus itself to aid in identification.
 - Stimulus driven.
- **Top-Down processes:**
 - Using context, previous knowledge, and expectation to aid in identification.
 - Knowledge driven.

Bayesian Inference

Bayes' Theorem describes an optimum method of combining bottom-up and top-down information.

Bayes' Theorem:

$$p(A|B)p(B) = p(B|A)p(A)$$

or

$$p(A|B) = p(B|A)p(A)/p(B)$$

$p(A|B)$ is the **conditional probability** of A given B (vertical bar “|” reads as “given”).

Bayesian Inference

An example of conditional probabilities.

The conditional probability that it is raining given that the pavement is wet is:

$$p(\text{rain} \mid \text{wet pavement}) < 1$$

because a wet pavement can be caused by many things (leaking pipes, dropped water bottles, etc).

The conditional probability that the pavement is wet given that it is raining is:

$$p(\text{wet pavement} \mid \text{rain}) = 1$$

because rain always wets the pavement.

Therefore, the two conditional probabilities are not necessarily equal

$$p(\text{rain} \mid \text{wet pavement}) \neq p(\text{wet pavement} \mid \text{rain})$$

Bayes' theorem gives the relationship between conditional probabilities.

Bayesian Inference

Bayes' theorem can be considered as a method for obtaining the information you need from the information you have.

In vision, we want to know $p(\text{object}_j \mid \text{Image}_i)$: the probability that object_j is present in the world given that image_i is on the retina.

Solving this is hard – it is an inverse problem

Bayesian Inference

Bayes' theorem can be considered as a method for obtaining the information you need from the information you have.

In vision, we want to know $p(\text{object}_j | \text{Image}_i)$: the probability that object_j is present in the world given that image_i is on the retina.

Solving this is hard – it is an inverse problem

However, we can calculate $p(\text{Image}_i | \text{object}_j)$: the probability of observing image_i given the 3D object_j.

Solving this is easier – it is a forward problem

Bayes' theorem provides a means of calculating $p(\text{object}_j | \text{Image}_i)$ since:

$$p(\text{object}_j | \text{Image}_i) = p(\text{Image}_i | \text{object}_j) p(\text{object}_j) / p(\text{Image}_i)$$

Bayesian Inference: nomenclature

$$p(\text{object}_j \mid \text{Image}_i) = p(\text{Image}_i \mid \text{object}_j) p(\text{object}_j) / p(\text{Image}_i)$$

posterior likelihood prior evidence

posterior: the thing we want to know (the probability of a particular object being present given the image).

likelihood: the thing we can calculate (the probability of the particular image being a projection of the particular object).

prior: the thing we know from prior experience (the probability that the particular object will be present in the environment)

evidence: the thing we can ignore, as it is the same for all possible interpretations of this image.

Bayesian Inference: example

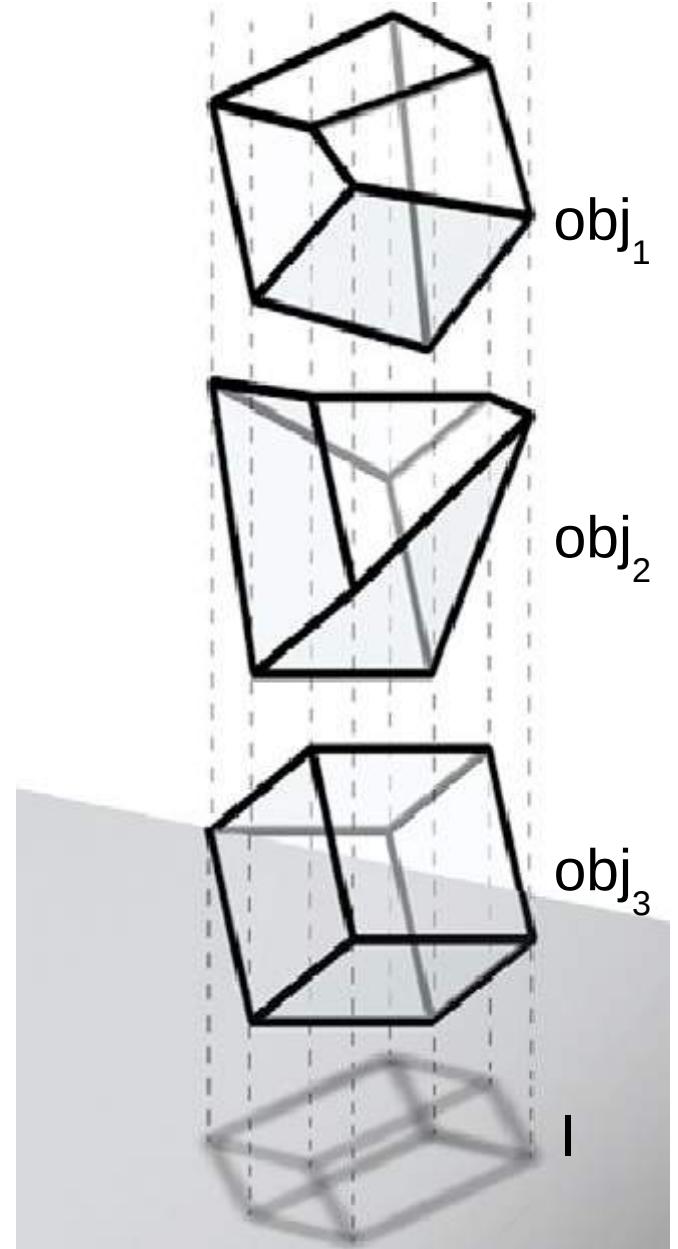
Each of $N=3$ possible objects can generate the observed image.

The probability of observing this image, I , is constant (make $p(I)=1$ for simplicity).

The likelihood $p(I|obj_j)$ is: “the probability of observing image I , given the 3D object obj_j ”.

If all $N=3$ objects could produce the same image with equal probability, their likelihoods are the same:

$$p(I|obj_1) = p(I|obj_2) = p(I|obj_3) = 0.09$$



Bayesian Inference: example

Thus, the image alone cannot be used to decide which of the three possible objects produced the image.

However, if our prior experience of 3D objects produces a higher expectation of cubes than irregular shapes, then the priors will be different: e.g. $p(\text{obj}_3) = 0.1$, $p(\text{obj}_2) = 0.01$, $p(\text{obj}_1) = 0.01$

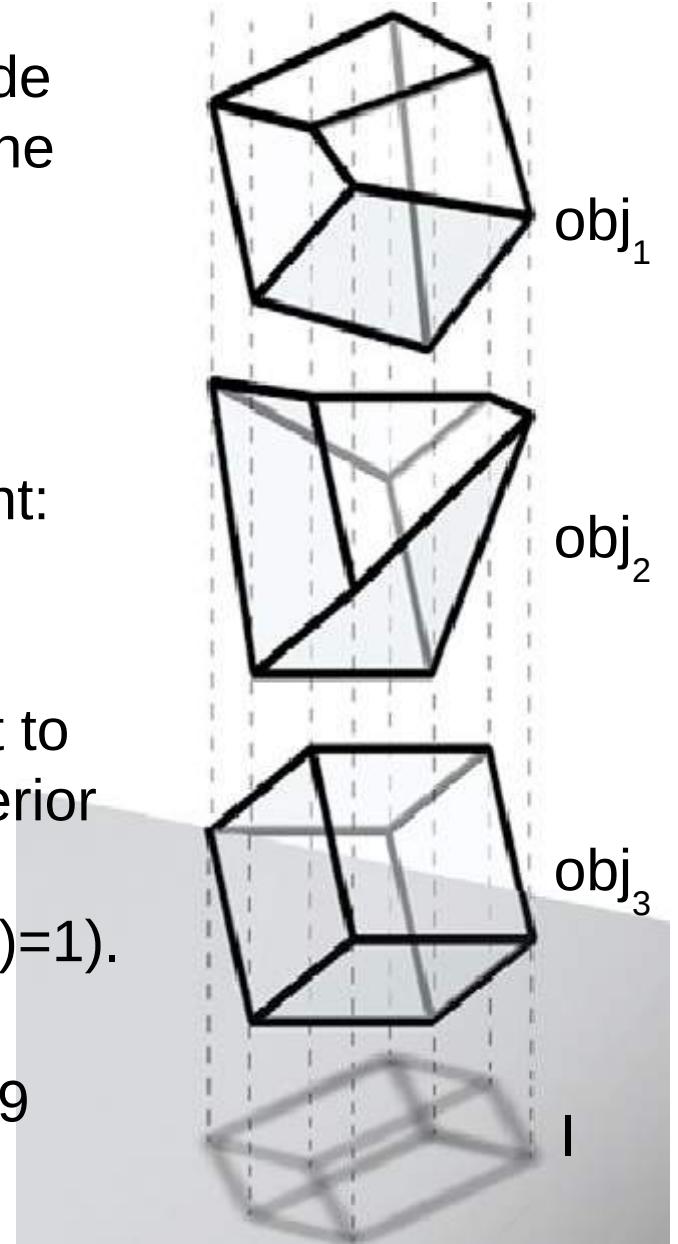
We can use the prior probability of each object to weight the known likelihood to obtain the posterior probability:

$$p(\text{obj}_j|I) = p(I|\text{obj}_j) p(\text{obj}_j) \text{ (assuming } p(I)=1).$$

Hence,

$$p(\text{obj}_1|I) = p(\text{obj}_2|I) = 0.09 \times 0.01 = 0.0009$$

$$p(\text{obj}_3|I) = 0.09 \times 0.1 = 0.009$$

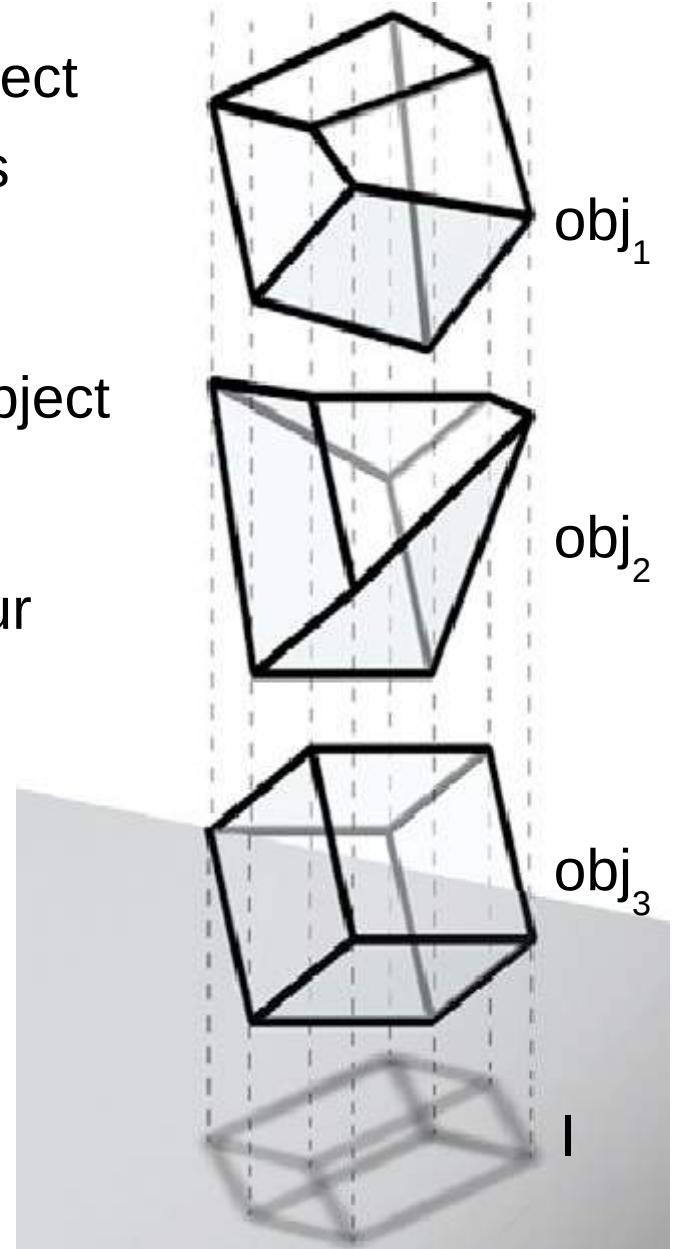


Bayesian Inference: example

The posterior $p(\text{obj}_j | I)$ is the probability that object obj_j is present in the world given that image I is on the retina.

The posterior probabilities thus tell us which object is most likely to have yielded image I .

In this example, the prior experience biases our interpretation of the image, so that we tend to interpret the image I as object obj_3 .



Bayesian Inference

Bayes rule shows how to combine current evidence, I , with knowledge gained from prior experience, $p(\text{obj}_j)$, to estimate the posterior probability $p(\text{obj}_j|I)$ that the hypothesis (obj_j) under consideration is true (e.g. that obj_j is the correct 3D object).

Need to compute posterior $p(\text{obj}_j|I)$ for all possible hypotheses in order to select that hypothesis with the largest posterior.

If we assume $p(I)=1$ then

$$\text{posterior} = \text{likelihood} * \text{prior}$$

Bayesian Inference

Alternatively, if we just want to determine the probability that an image contains a particular object or not, we can use the following formulation:

$$\frac{p(\text{object}_j | \text{image}_i)}{p(\text{not object}_j | \text{image}_i)} = \underbrace{\frac{p(\text{image}_i | \text{object}_j)}{p(\text{image}_i | \text{not object}_j)}}_{\text{likelihood ratio}} \cdot \underbrace{\frac{p(\text{object}_j)}{p(\text{not object}_j)}}_{\text{prior ratio}}$$

posterior ratio

likelihood ratio

prior ratio

Bayesian Inference: example



$$p(\text{image} \mid \text{zebra}) = 0.07$$

$$p(\text{zebra}) = 0.01$$

$$p(\text{image} \mid \text{no zebra}) = 0.0005$$

$$p(\text{no zebra}) = 0.99$$

$$\frac{p(\text{zebra} \mid \text{image})}{p(\text{no zebra} \mid \text{image})} = \frac{p(\text{image} \mid \text{zebra})}{p(\text{image} \mid \text{no zebra})} \cdot \frac{p(\text{zebra})}{p(\text{no zebra})}$$

$$= \frac{0.07}{0.0005} \frac{0.01}{0.99} = 1.41$$

>1 so zebra

Bayesian Inference: example



$$p(\text{image} \mid \text{zebra}) = 0.003$$

$$p(\text{zebra}) = 0.01$$

$$p(\text{image} \mid \text{no zebra}) = 0.85$$

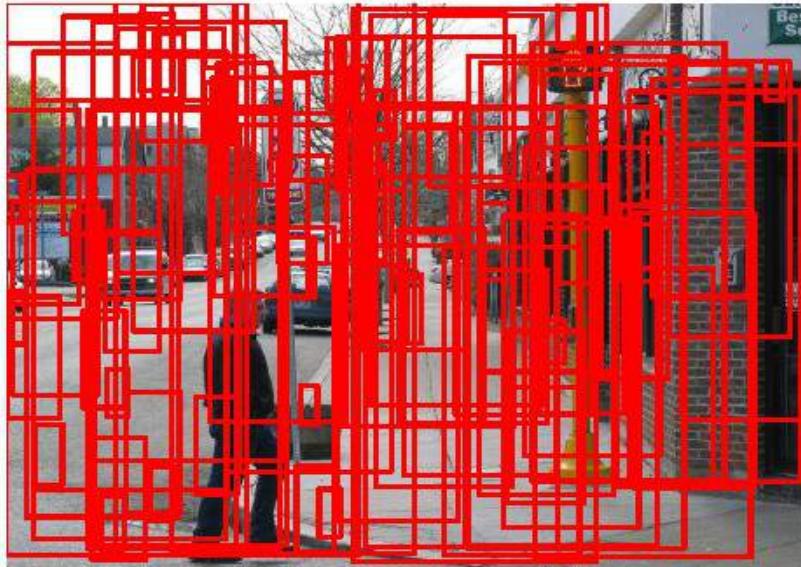
$$p(\text{no zebra}) = 0.99$$

$$\frac{p(\text{zebra} \mid \text{image})}{p(\text{no zebra} \mid \text{image})} = \frac{p(\text{image} \mid \text{zebra})}{p(\text{image} \mid \text{no zebra})} \cdot \frac{p(\text{zebra})}{p(\text{no zebra})}$$

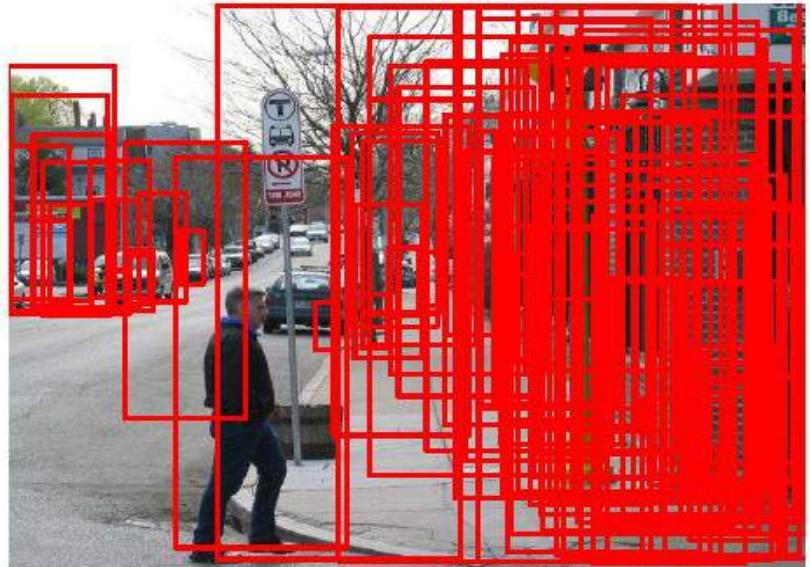
$$= \frac{0.003}{0.85} \frac{0.01}{0.99} = 0.000036$$

<1 so not zebra

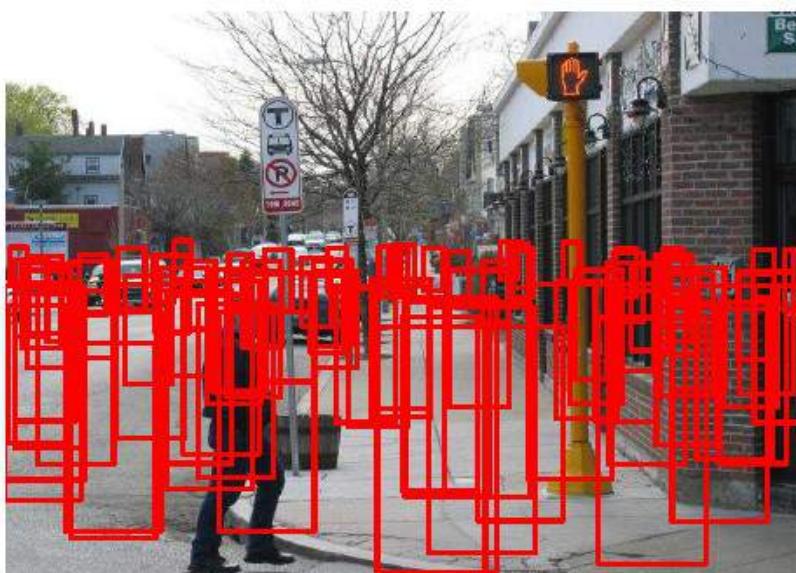
Bayesian Inference: example



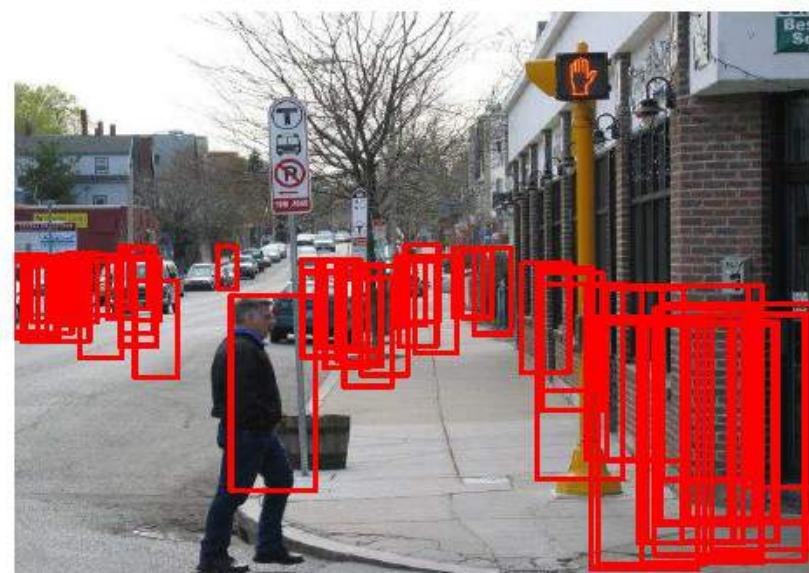
(b) $P(\text{person}) = \text{uniform}$



(d) $P(\text{person} \mid \text{geometry})$



Co₁ (f) $P(\text{person} \mid \text{viewpoint})$



(g) $P(\text{person} \mid \text{viewpoint, geometry})$ ₆₂

Bayesian Inference

Bayesian inference can be seen as a method of solving the ill-posed, inverse problem of vision (see introductory lecture)

Vision is an inverse problem – we know the pixel intensities (the outcomes) and want to infer the causes (i.e. the objects in the scene, etc.).

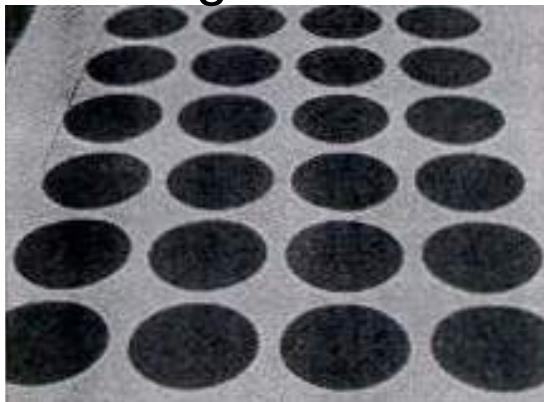
Vision is ill-posed as there are usually multiple solutions (i.e. multiple causes that could give rise to the same outcomes).

In order to compensate, the perceptual systems make use of assumptions, constraints or priors about the nature of the physical world.

Bayesian Inference

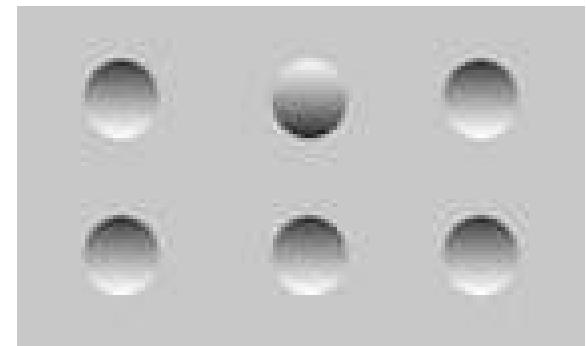
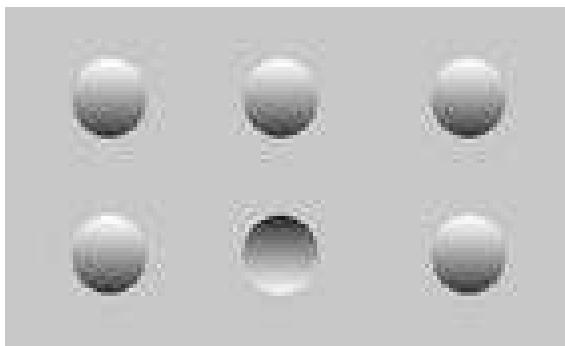
Prior: Texture is circular and homogeneous

Infer: shape/depth



Prior: Light from above

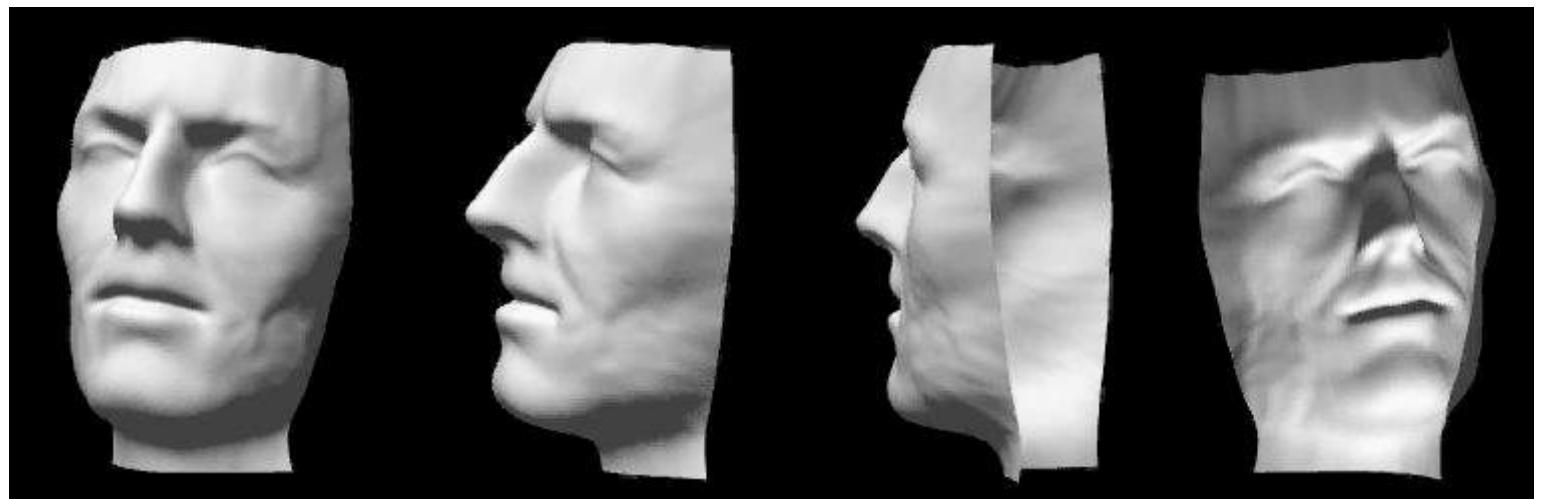
Infer: depth



Bayesian Inference

Prior: faces are convex

Infer: shape/depth



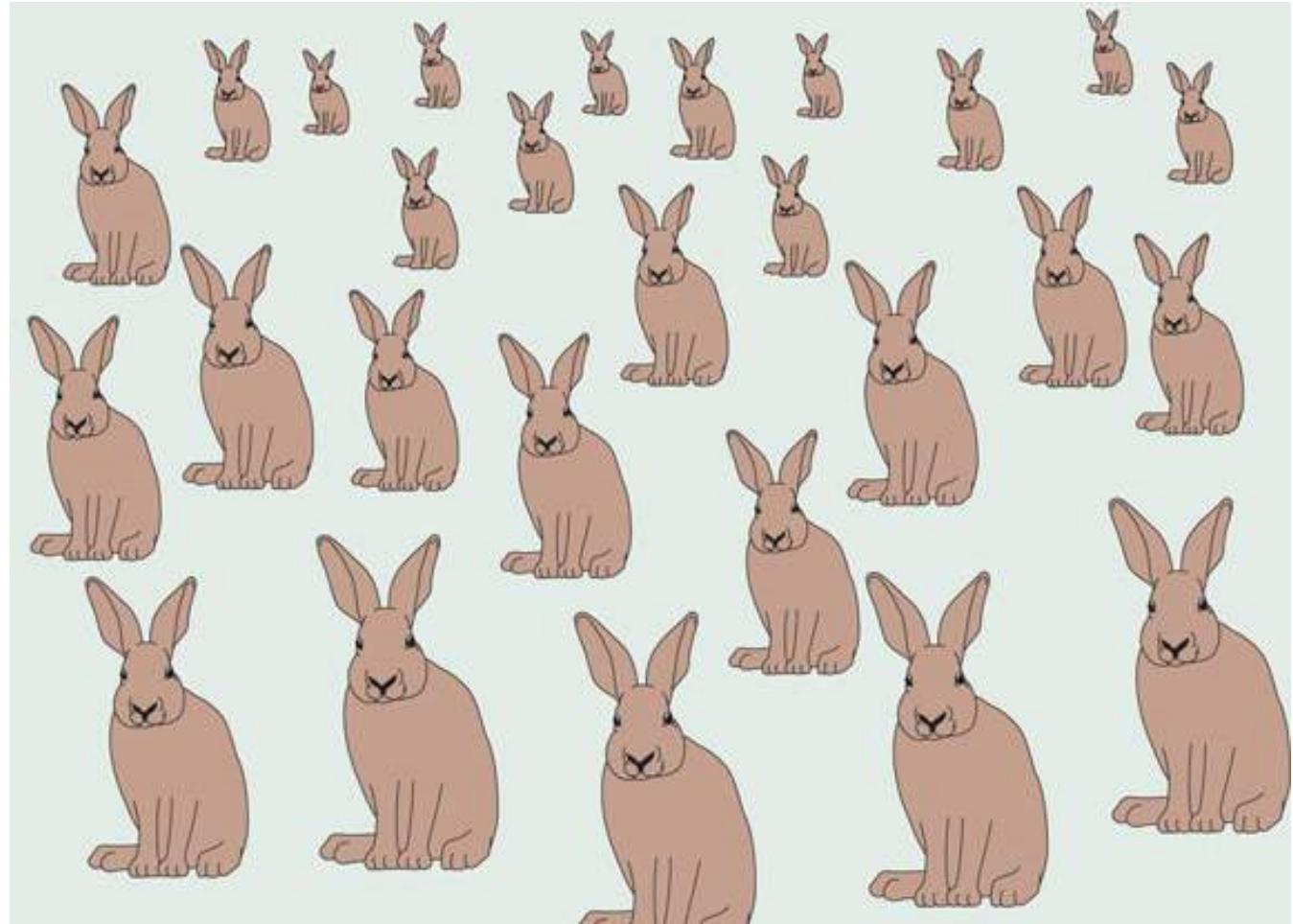
Convex face appears convex
if assume light comes from
above

Concave face still appears
convex, but only if assume
light now comes from below

Bayesian Inference

Prior: size is constant

Infer: depth



Bayesian Inference

Prior: neighbouring features are related

Infer: grouping



Prior: similar features are related

Infer: grouping



Prior: connected features are related

Infer: grouping

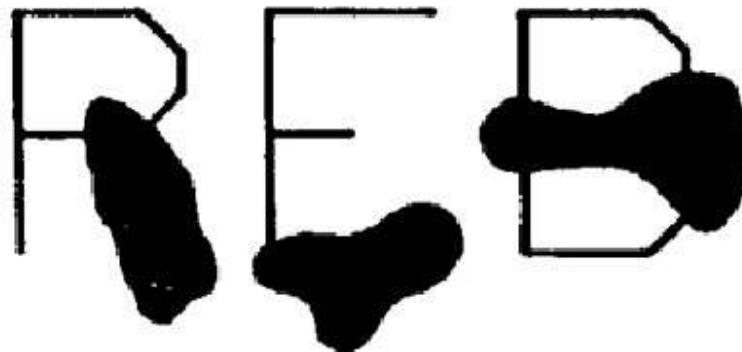


Bayesian Inference

Prior: strings of letters form words

Infer: letter identity

T A E C A T



Bayesian Inference

Prior: knowledge about image content

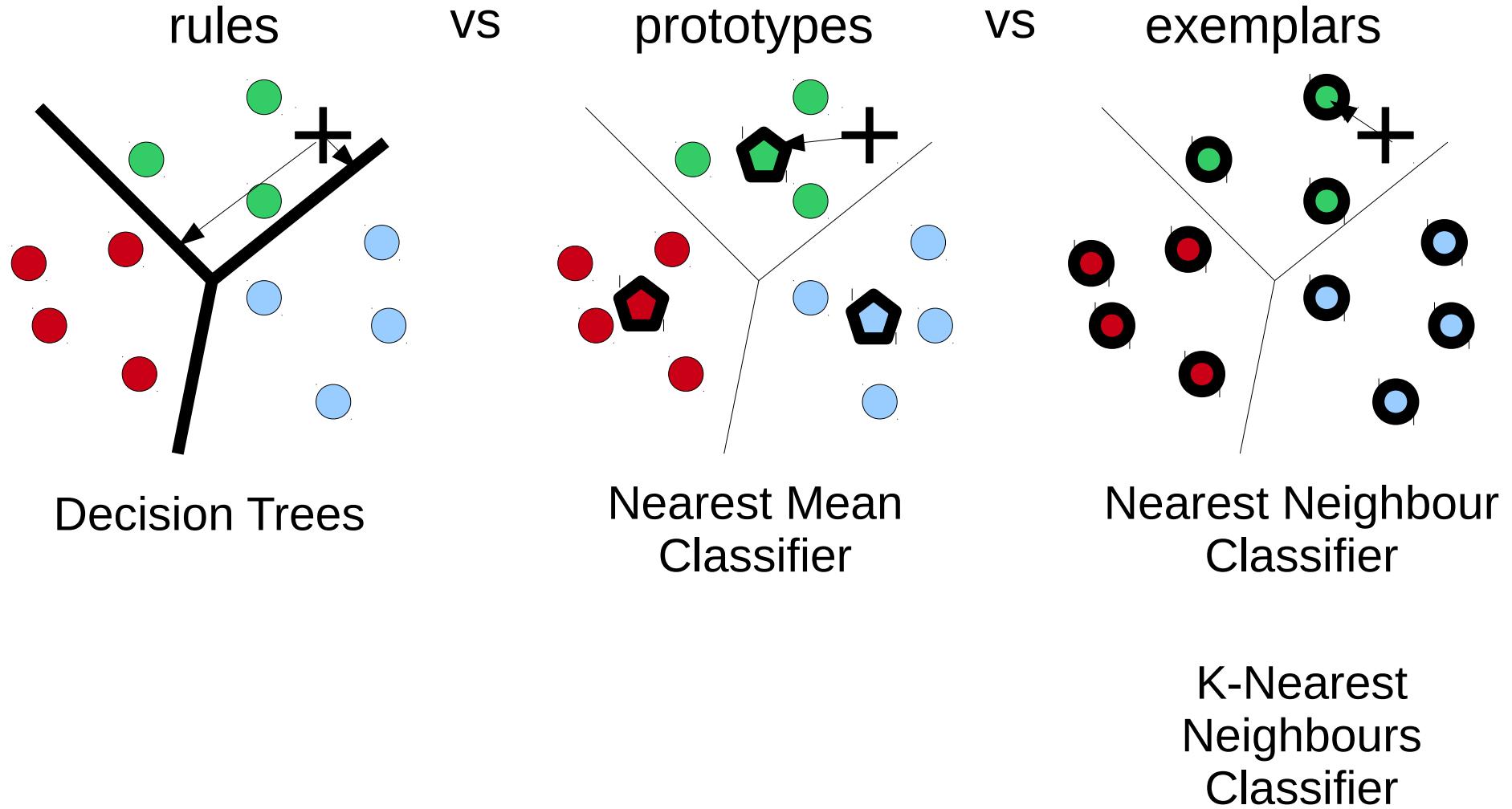
Infer: object identity



We are back where we started in lecture 1!

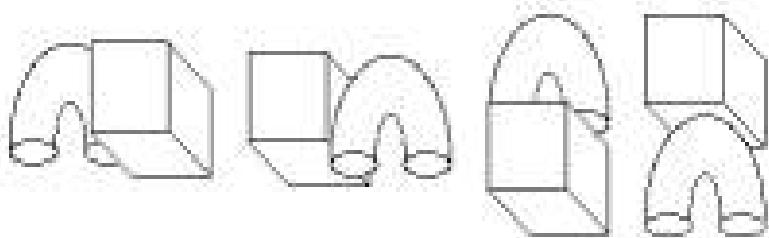
Summary

psychology



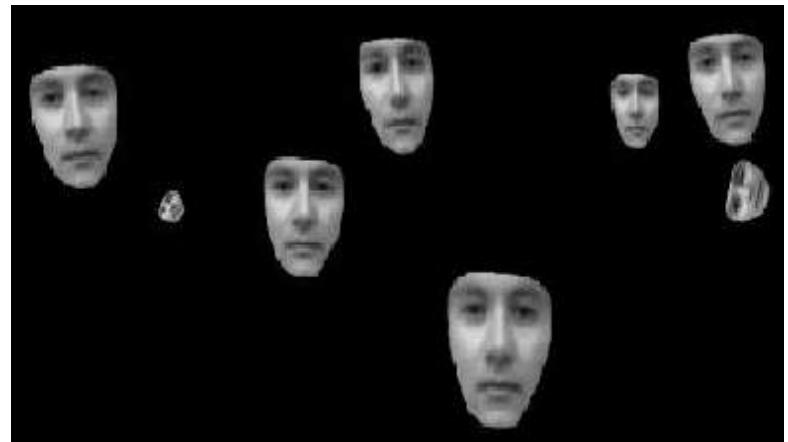
Summary

object-based (3D)
e.g. recognition by
components



vs

image-based (2D)
e.g. template matching



configural (global)



vs

featural (local)



Summary

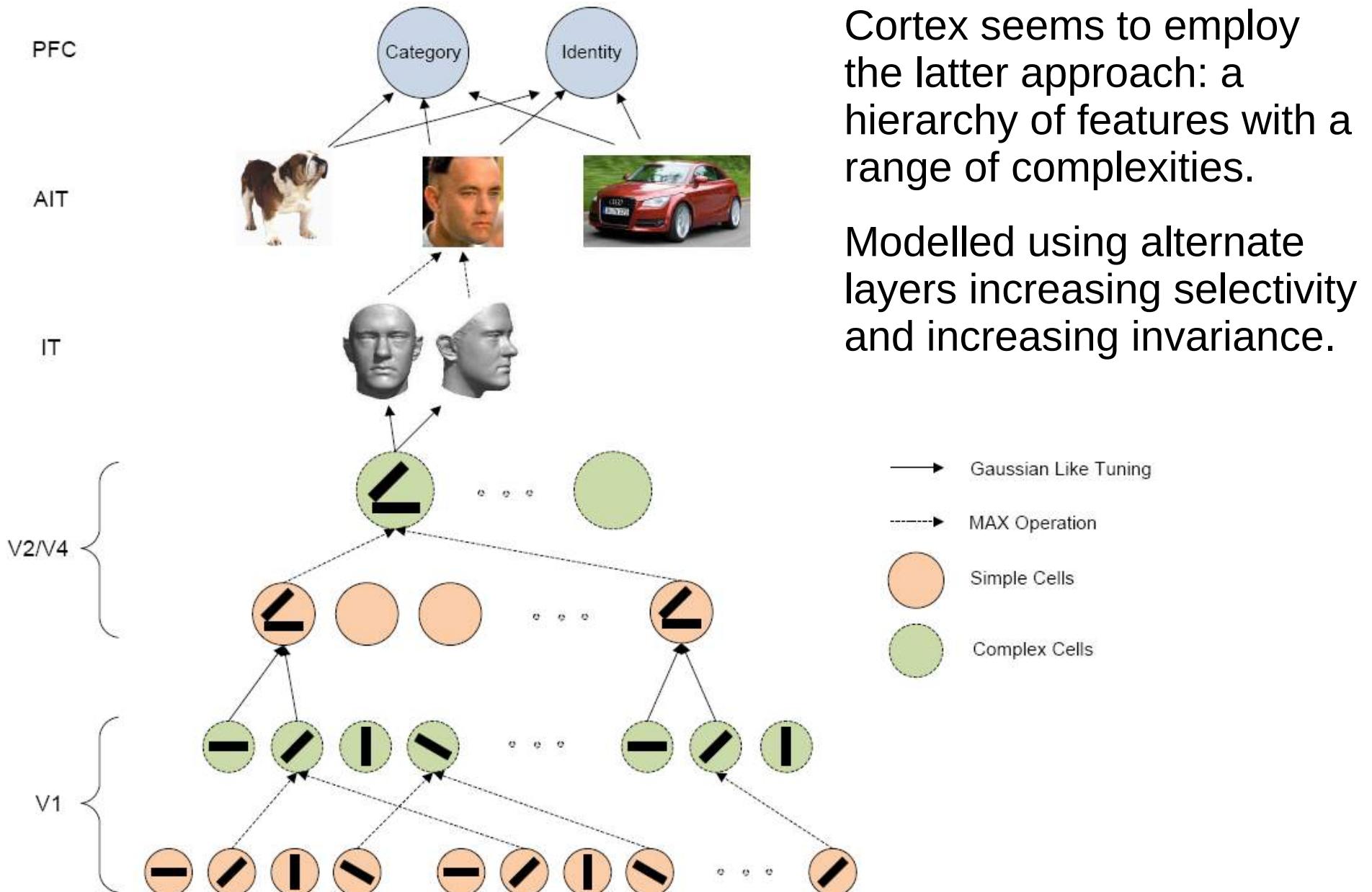
Local (featural) and global (configural) representations have complementary advantages and disadvantages

- simple (local) features generate many false positives:
 - fail to distinguish objects with similar features in different arrangements,
 - fail to deal with clutter
- complex (global) features generate many false negatives
 - fail to deal with occlusion
 - fail to deal with viewpoint changes and within class variation

Solutions:

1. use features of intermediate complexity
2. use a hierarchy of features with a range of complexities

Summary



Summary

- **Bottom-Up processes**
 - Using the information in the stimulus itself to aid in identification
 - Stimulus driven
 - Discriminative
- **Top-Down processes**
 - Using context, previous knowledge, and expectation to aid in identification
 - Knowledge driven
 - Generative

Summary

$$p(\text{object}_j \mid \text{image}_i) = \underbrace{p(\text{image}_i \mid \text{object}_j)}_{\text{likelihood}} \underbrace{p(\text{object}_j)}_{\text{prior}} / \underbrace{p(\text{image}_i)}_{\text{evidence}}$$

$$\frac{p(\text{object}_j \mid \text{image}_i)}{p(\text{not object}_j \mid \text{image}_i)} = \underbrace{\frac{p(\text{image}_i \mid \text{object}_j)}{p(\text{image}_i \mid \text{not object}_j)}}_{\text{likelihood ratio}} \cdot \underbrace{\frac{p(\text{object}_j)}{p(\text{not object}_j)}}_{\text{prior ratio}}$$

- Discriminative methods model the posterior
- Generative methods model the likelihood and prior