Session 7: Computer Vision 1

Dr John Fawcett

Goals of Computer Vision

- Scene understanding
- Extract information from an image
- Extract information from a video

Colour Inference

Humans use knowledge of the illumination colour to infer object surface colours

- Specular reflections in the colour of the light
- Diffuse reflections in the colour of the surface
- Might have an ambient illumination term too

Solution to lighting equations is not unique without knowledge of light position + colour

- Different numbers or colours of lights in scene
- Direct or indirect illumination of scene

Texture Inference

Separate images at texture boundaries

- Need to recognise textures
- Textures have quasi-periodicity and can be exploited by spatially localised Fourier transforms
- Unify regions with the same textures
- Disjoint objects at texture boundaries

Inferring surface from texture also possible

- Foreshortening of a quasi-periodic pattern suggests depth into the scene

Edges

Hugely important clues into scene structure!

Need to detect edges – a mathematical challenge

Noise is a huge issue

Need to convert *signal* into *symbols*

- Scene understanding
- Recognition
- Classification
- Identification

Edges

Need to segment images at the edges

- Yields scene descriptions

Scale-space pyramid to declutter and draw out the key structural edges

- Blur and down-sample the image at a variety of resolutions
- Edge detect each down-sampled, blurred image
- Fingerprint identifies objects
- Strong edges help with *Correspondence Problem*

Metaphysical Knowledge

Humans have metaphysical knowledge about the colours of objects

- E.g. bananas look yellow under many different lighting conditions!

Humans have metaphysical knowledge about object interactions

- Constrains 3D scene reconstruction
- E.g. tables do not float in the air

Metaphysical Knowledge

- We need a way to *encode* the knowledge in a mathematical notation
- We need a way to *incorporate* the knowledge into the mathematical reasoning
- Bayesian Priors often helpful

Learning from Human Vision

No cognitive penetrance!

What algorithm does human vision use?

- It’s highly parallel
- It’s goal-driven
- It’s an approximation
- It’s influenced by past experience
- It’s influenced by what we feel, hear, smell
- It’s real-time

Learning from Human Vision

Vision is reverse graphics!

- Brain guesses what’s in the scene

Stochastic

- Parallel

Brain renders those scenes into images

- Like ray-tracing

Compare ray-traced images to data from eyes

Update scene description using best match

Iterate

Learning from Human Vision

Use knowledge of the real world to skew the distribution of scenes attempted

- E.g. no floating tables
- Incorporates metaphysical knowledge using Bayesian Priors

Extends to video: prefer explanations with few changes since the last video frame

Edge Detection

Consider the 2D filter [+1,-1]

Apply to an image by direct convolution

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Generates an image one column narrower

Detects vertical edges

Different response polarity for dark—light edges vs light—dark edges

Edge Detection

0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1

X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0

--> Convolve with [+1,-1] -->

Edge Detection

X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0
X	0	0	0	0	1	0	0	0	0

X	X	0	0	0	1	-1	0	0	0
X	X	0	0	0	1	-1	0	0	0
X	X	0	0	0	1	-1	0	0	0
X	X	0	0	0	1	-1	0	0	0
X	X	0	0	0	1	-1	0	0	0
X	X	0	0	0	1	-1	0	0	0
X	X	0	0	0	1	-1	0	0	0
X	X	0	0	0	1	-1	0	0	0
X	X	0	0	0	1	-1	0	0	0
X	X	0	0	0	1	-1	0	0	0

--> Convolve with [+1,-1] --> Convolve with [+1,-1] -->

Edge Detection

Convolving with a large image, twice, is slow

Pre-compute filter applied to filter

Apply result to image once

[a,b,c,d] * [+1,-1] = [X, -a+b, -b+c, -c+d]

[X, -a+b, -b+c, -c+d] * [+1,-1] =
[X, X, a-2b+c, b-2c+d]

Just convolve with [-1, +2, -1]

Edge Detection

Doing this for real:

- Extend into 2 dimensions
- Apply blurring to reduce noise
- Apply using multiplication/convolution duality of Fourier transform

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

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

Linear Operators

Linear operators have…

- *Additivity* property:
  if F(u) ⇔ f(x) and G(u) ⇔ g(x)
  then F(u)+G(u) ⇔ f(x)+g(x)
- *Proportionality* property:
  if F(u) ⇔ f(x)
  then A.F(u) ⇔ A.f(x)

Linear Operators

- First differentiates a blurred image
- Second blurs a differentiated image
- Third differentiates the blurring operator and applies the result to the image.

- All three yield same result!
- Third is much cheaper
  - Loops over large image once!

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

$$G_\sigma(x,y) * \nabla^2 I(x,y)$$

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

Gradient Vector Fields

Local spatial derivative of the image, I(x,y)

Used by Canny edge detector:

- Gaussian blur the image
- Edge detect
- Apply line thinning
- Double-threshold edges: strong, weak, rejected
- Connected components analysis
- Keep strong edges and weak edges that connect to strong edges

Active Contours

- Fit a deformable boundary to the set of pixels where edges were detected
- Signal to symbol conversion!
- Parameterised piecewise spline template
- Define *Internal Energy*: how deformed the template shape has to be in order to fit data
- Define *External Energy*: how well the deformed template shape fits the data

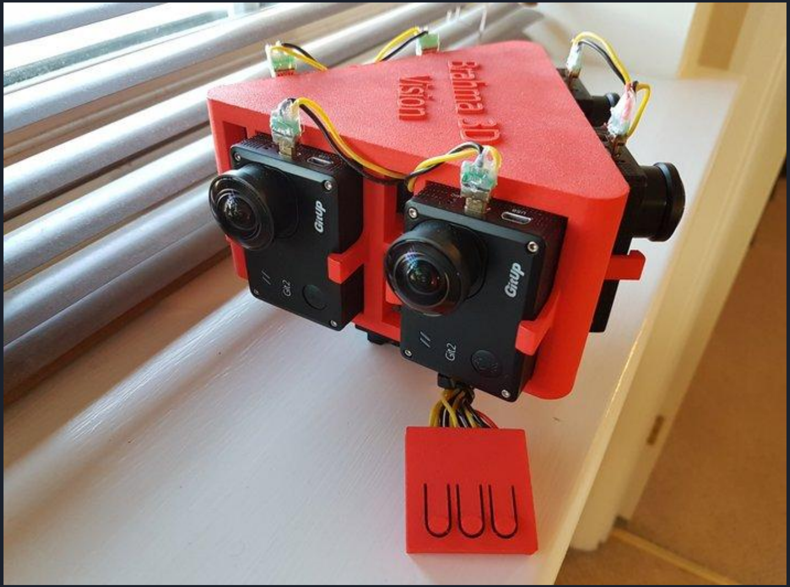
Active Contours

- Minimise weighted sum of internal and external energy by adjusting parameters of the model
- Often high-dimensionality problem
- Methods used
  - Gradient descent
  - Simulated annealing
  - Partial differential equations
- Often slow and numerically unstable!

Depth Inference

Suppose we have a pair of identical cameras

Can we infer depth?



Depth Inference

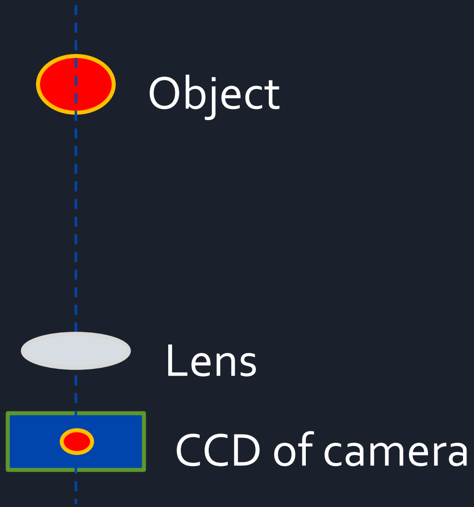
Obvious algorithm:

- Take two pictures at the same time
- Find the object of interest in both images
- Compare where it appears in the images
- Infer depth into the scene

What does the maths look like?

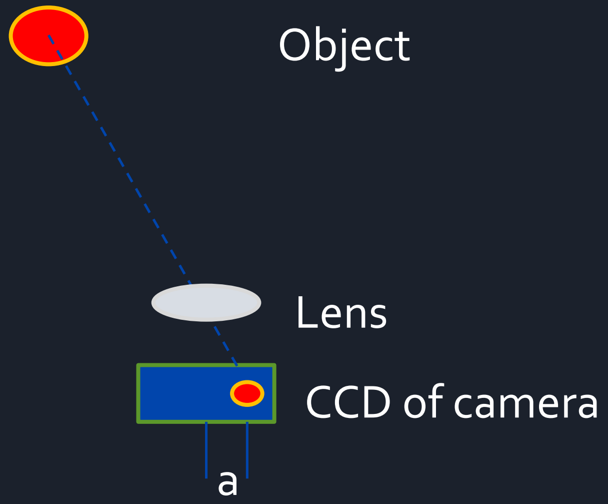
Depth Inference

Objects in front of a camera appear in the centre of a captured image



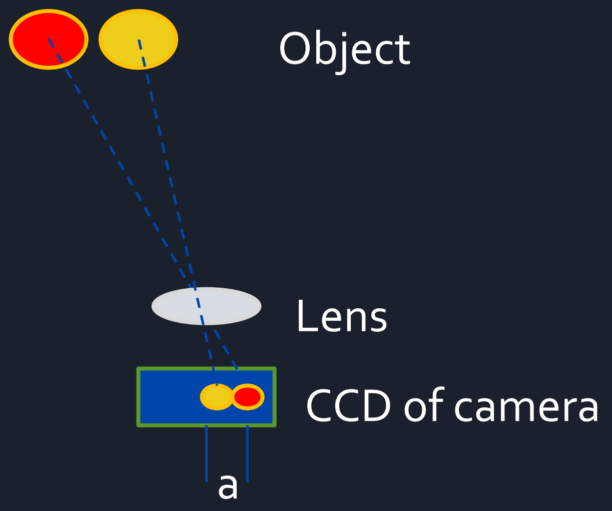
Depth Inference

Objects to one side are displaced from the centre of an image by distance ‘a’



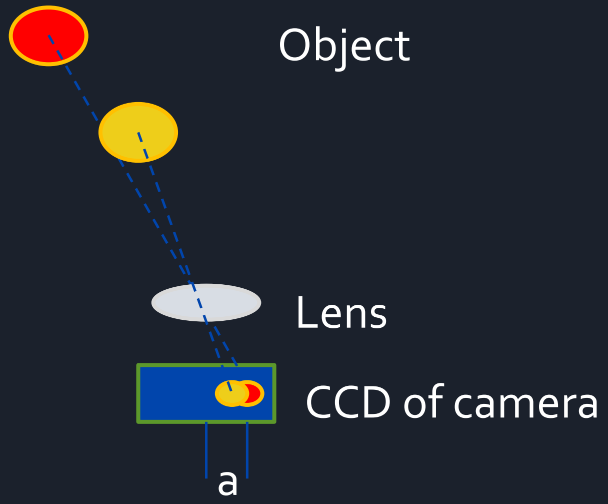
Depth Inference

Displacement in the image is proportional to the displacement in the scene



Depth Inference

Displacement in the image is inversely proportional to the depth



Depth Inference

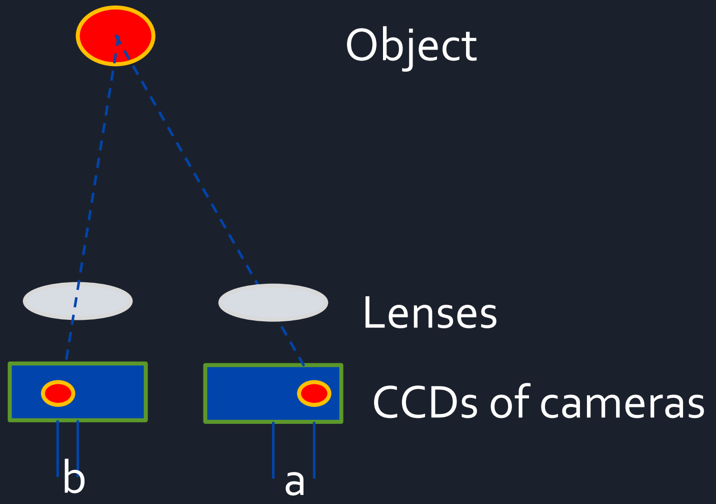
We have two unknowns: lateral position in the scene and depth

Without knowing either depth or lateral displacement, we cannot work out the other

Simultaneous equations derived from the images taken by two cameras allows us to deduce the depth…

Depth Inference

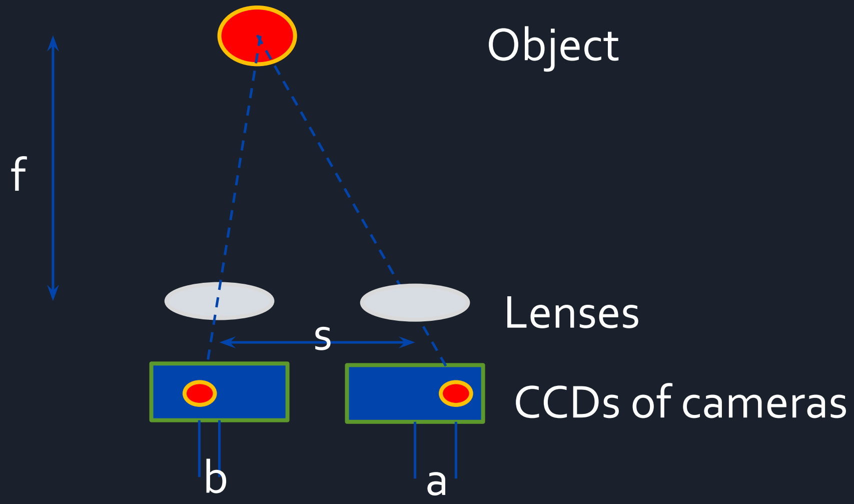
Object is seen at displacement ‘a’ in one image and ‘b’ in the second image



Depth Inference

Using similar triangles, the depth, d, is given by

$$d = \frac{fs}{a+b}$$



s: camera separation distance
f: focal length
d: depth of object into scene (from the plane of the camera lenses)
a, b: displacements in the images

Calibration

- Our cameras need to be calibrated very accurately
  - Relative position in 3 dimensions
  - Relative rotation in 3 dimensions
  - Focal length
- Any errors in these values are amplified by the division
  - a and b are small distances compared to d

$$d = \frac{fs}{a + b}$$

Now it’s time for you to have a go

Exercise sheet!