CSC 355 - Artificial Intelligence

Computer Vision

Hans Gellersen

Objectives & Credits

Objectives

- To provide a glimpse of what computer vision is about
- To give an understanding of image processing for computer vision
- Focus on early processing of images and the determination of structure: edges, lines, shapes.

Credits

- All material used in this lecture is based on course material of Dr David Young at University of Sussex
- Link to the complete course: http://www.cogs.susx.ac.uk/courses/compvis/index.html
  - We will cover some material from lectures 1-7
- See also the Sussex Computer Vision Teach Files: http://www.cogs.susx.ac.uk/users/davidy/teachvision/vision0.html
  - Highly recommended in addition to lecture notes

Overview

- What is Computer Vision?
- Exercise: Image Structure for 3D
- Image Filtering, Convolution
- Scale Space
- Edge Detection
- The Hough Transform
- Active Contours

What is Computer Vision

To do with seeing

- using information mediated by light in order to interact successfully with the environment

As much to do with biological systems as with computers, but there are many different approaches:

- How do people and animals see?
- How can we make useful robots that see?
- What are the general computational structures that underly vision?
- How do we reconstruct the 3rd dimension from 2-D images?
- How can we build machines to solve specific tasks involving vision?

Very interdisciplinary

- Artificial Intelligence, Computer Science, Engineering, Psycholog, Neuroscience, Mathematics

We focus on computational methods for finding structure in images

Applications

License Plate Recognition

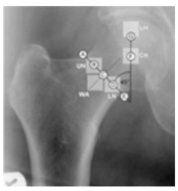
- London Congestion Charge
- http://www.cclondon.com/imagingandcameras.shtml

• **Surveillance**

- Face Recognition
- Airport Security (People Tracking)

• **Medical Imaging**

- (Semi-)automatic segmentation and measurements



What Vision Programs do …

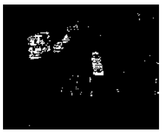
What do computer vision programs actually *do*?

- Image processing, e.g. edge detection:

- Image processing is *not* computer vision — but it is an important part of it.
- Work *bottom-up* to find structure in images.

What Vision Programs do …

- Motion detection:



- Finding shape:

The "bottom-up" approach

Work bottom-up to find structure

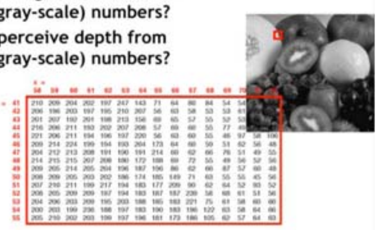
- Start from a grey-level array (the image, in effect)
  - colour is usually ignored: not important for finding structure
- Primal sketch: edges, groupings of edges
- 2.5-D sketch: surface depth and orientations
- 3-D model: object shapes and relationships

In some sense, the 3-D model is taken as the goal of the visual processing.

It can be used for matching against a database of object shapes to achieve object identification.

The problem of 'inverse graphics'

- 'Goals' of Computer Vision
  - how can we recognize fruits from an array of (gray-scale) numbers?
  - how can we perceive depth from an array of (gray-scale) numbers?
  - …
- the problem of 'inverse graphics'…



Computer Vision Goals

The bottom-up approach

- 'traditional' but still useful for framework for understanding what vision programs do

Better goal:

- Produce systems that enable successful interaction with the environment (rather than aiming at a particular representation)
  - navigating an autonomous vehicle along a road and past obstacles
  - recognising human gestures and movements for computer control
  - etc
- Work top-down and hypothesis-driven: start with an assumption of what the system (e.g. robot) sees, and test whether the image matches the hypothesis
- Dynamic vision: change and motion ('optical flow') are often more important than recognising shape or inferring the 3rd dimension

Overview

- What is Computer Vision?
- Exercise: Image Structure for 3D
- Image Filtering, Convolution
- Scale Space
- Edge Detection
- The Hough Transform
- Active Contours

Exercise

Look at images of 3D scenes

What kinds of information does the image contain that might allow a computer program to reconstruct the 3D scene

Image Structure for 3D

Linear Perspective
- Look for fans of straight lines diverging from a vanishing point.
- Assume the scene actually contains parallel lines in 3-D.

Texture gradient
- Look for systematic changes in image size of similar objects.
- Assume that things with similar appearance have a fairly uniform size in the scene.

Height in image
- Higher up in the image often means further away.
- Assume that the scene lies on a ground plane.

Image Structure for 3D

Size of known objects
- The image size of a recognised object of known dimensions gives its distance.
- Assume that you're seeing the real thing, not a model for example.

Shading
- Image brightness gives information about surface orientation relative to light source.
- Assume that the brightness variation is not intrinsic to the surface.

Foreshortening
- The image shape of a surface patch depends on its orientation — e.g. circles foreshorten to ellipses.
- Assume known true shape, or symmetry, of contour.

Image Structure for 3D, cont.

Focus
- Things nearer or farther than the point of focus may be blurred.
- Assume that real objects generally have sharp boundaries.

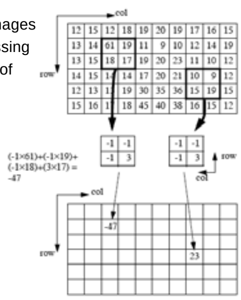
Occlusion
- The images of nearer things cut across the image structure of farther things.
- Assume that the edges of objects are not accidentally lined up in the image ("general viewpoint" assumption).

Overview

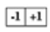
- What is Computer Vision?
- Exercise: Image Structure for 3D
- Image Filtering, Convolution
- Scale Space
- Edge Detection
- The Hough Transform
- Active Contours

Convolution

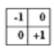
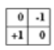
- A class of local operations on images
- Central to modern image processing
- The basic idea is that a window of some finite size and shape is scanned across the image.
- The window with its weights is called the convolution kernel, filter, or mask
- The output pixel value in the filtered image is the weighted sum of the input pixels within the window



Simple Differencing Masks

- Horizontal Differences 

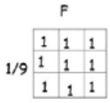
- Vertical Differences 

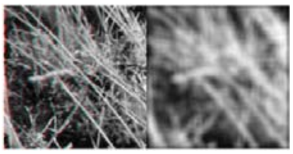
- Diagonal Differences  

A Simple Smoothing Filter

Average Filter

- Mask with positive entries that sum to 1
- replaces each pixel with an average of its neighborhood
- if all weights are equal, it is called a BOX filter

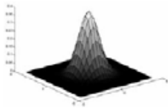


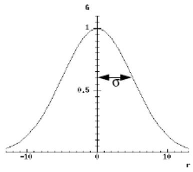


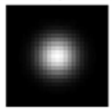
Not particularly good because its point-spread function has sharp edges

Gaussian Filter

- A more effective smoothing mask falls off gently at the edges
- The Gaussian mask is rotationally symmetric
- Weights nearby pixels more than distant ones
- Width of the mask is described by a parameter called σ (sigma)







Smoothing with a Gaussian

Overview

- What is Computer Vision?
- Exercise: Image Structure for 3D
- Image Filtering, Convolution
- Scale Space
- Edge Detection
- The Hough Transform
- Active Contours

Scale space

Scale

- The Gaussian mask effectively removes any texture with a scale smaller than the mask dimensions.
- Small-scale texture is said to have a high *spatial frequency*. Smoothing removes this, leaving low spatial frequencies.

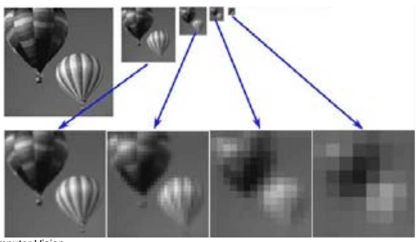
Images can then be considered as structures in scale space

- Storing an image at different scales (we will show why that's useful)
- The scale is defined by the smoothing parameter σ
- A property of the image is then considered to be a function not just of position (x,y) but also of scale σ (i.e. "look at the same point at different levels of smoothing")

Gaussian Pyramid

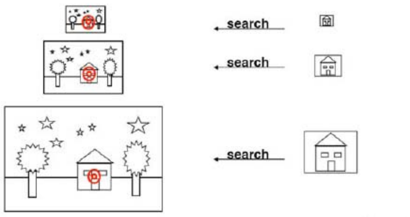
Resolution Pyramid

- At a bigger scale (i.e. larger σ) , there is no need to retain all the pixels. Instead the image can be stored in a resolution pyramid.

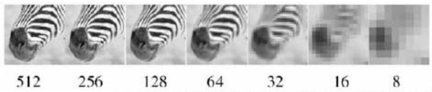


Scale Space: Motivation

Search across Scales



Scale Space: Another Example



- a bar
  - in the big images is a hair (on the zebra's nose)
  - in smaller images, a stripe
  - in the smallest image, the animal's nose

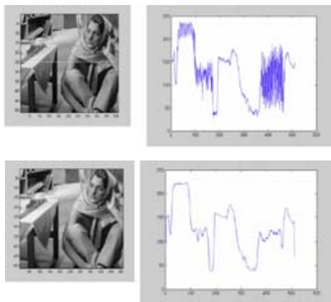
Overview

- What is Computer Vision?
- Exercise: Image Structure for 3D
- Image Filtering, Convolution
- Scale Space
- Edge Detection
- The Hough Transform
- Active Contours

What are Edges?

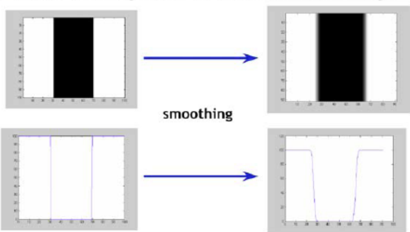
Significant changes in grey-level

- Actual 1D Profile (line 250)
- 1D Profile with a Gaussian

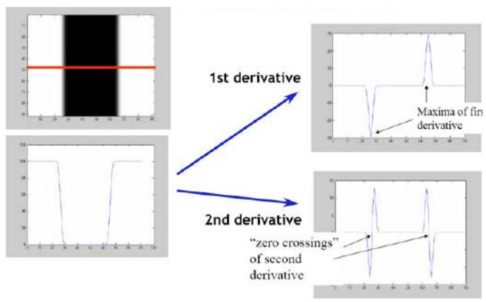


What are Edges?

- **edges:**
  - correspond to fast changes
  - where the magnitude of the derivative is large

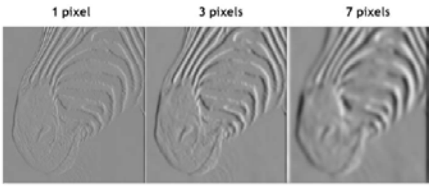


Edges and Derivatives



Edge detection and scale

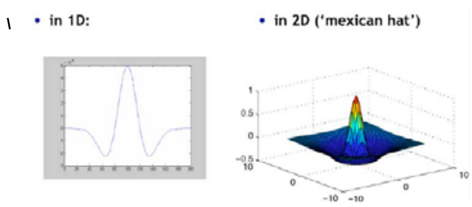
- The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered
  - note: strong edges persist across scales



Computer Vision HWG 31

Second Derivative of Gaussian

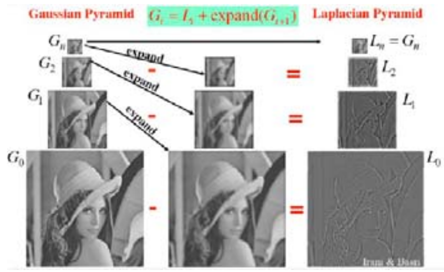
The *Laplacian of the Gaussian* Mask



Computer Vision HWG 32

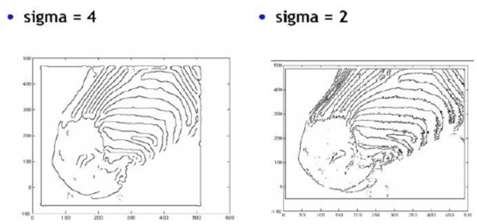
Laplacian Pyramid

Approximating the Laplacian with *Difference of Gaussians* at different scales



Computer Vision HWG 33

Edges at different scale



Computer Vision HWG 34

Overview

- What is Computer Vision?
- Exercise: Image Structure for 3D
- Image Filtering, Convolution
- Scale Space
- Edge Detection
- The Hough Transform
- Active Contours

Computer Vision HWG 35

Finding shapes

So far

- *Local* operations on images
- Finding basic structure (edges at different scale)

Next

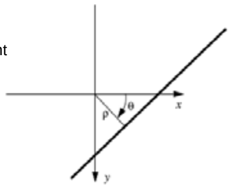
- Algorithms for obtaining extended image structure using more *global* operations on the image
- The Hough Transform
  - Finding evidence for well-defined parametrised geometrical shapes (circles, lines, etc.)
- Active Contours
  - Finding shapes which have certain properties that we can specify (other than their geometry)

Computer Vision HWG 36

Hough Transform

Hough Transform for straight lines

- A straight line can be described by two parameters (we are not at present interested in its end points):
  - θ (theta) is the angle from the x axis to the perpendicular to the line
  - ρ (rho) is the distance of the line from the origin, measured along its perpendicular.

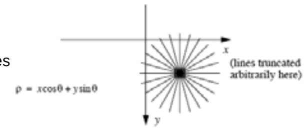


Parameter Space

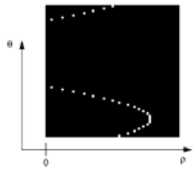
- Any line in the plane can be represented by a pair of values for ρ and θ.
- Thus each point in the (ρ,θ) parameter space corresponds to a single line, and vice versa.

Hough Transform

- A given point feature lies on many different lines
- It has a *vote* for all the lines it might lie on



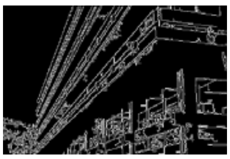
- In parameter space, all votes are accumulated
- The space is also called *accumulator array*
- The cells are often called *bins*.



Hough Transform

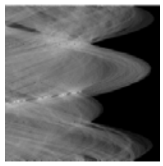
- A scene with some prominent straight lines

- And the edges found by an edge detection algorithm
- Each white pixel votes for 300 possible lines through it …



Hough Transform

- … to produce the accumulator array …



- from which the 18 largest peaks give the strongest lines:

General Hough Transform

The Hough Transform can be applied to any parametrised shape

- Shapes described by equations (lines, ellipses, circles, etc)
- Shapes described by tables, with orientation, position and maybe scale as parameters to be found
- Shapes that deform depending on their position (e.g. for finding iris position in eye images)

- cf. David Young's lecture notes for HT for circles

Overview

- What is Computer Vision?
- Exercise: Image Structure for 3D
- Image Filtering, Convolution
- Scale Space
- Edge Detection
- The Hough Transform
- Active Contours

Active Contours

Active contours, or deformable contours, or *snakes*

- Hough transforms find evidence for well-defined parametrised geometrical shapes

- But often what is needed is to find extended shapes that do not have a concise geometrical description, but which have certain properties that we can specify.

Motivation

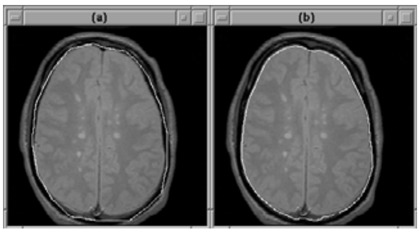
Why active contours?

- The shape of many objects is not easily represented by rigid primitives. For example:
  - Natural objects, such as bananas, have similar recognizable shapes. But no two bananas are exactly the same.
  - In medical imaging, objects are similar but not exact. An exact representation of a vein's shape, for example, cannot be given.
  - Some objects, such as lips, change over time.

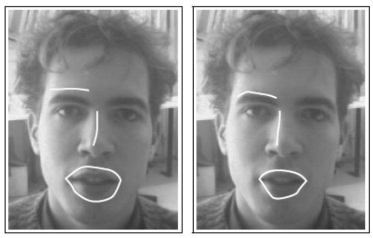
Idea of a snake

- Draw a rough outline (a “snake”) around a shape
- The snake then closes in on the shape to delineate it

Example: Medical Imaging



Example: Facial Features

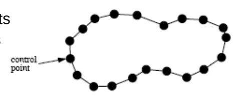


Initial configuration Final configuration

Snakes

What is a snake?

- A contour in the image plane
- Defined by a set of control points
- Deforms to fit certain properties



The snake’s position and shape is made to evolve to satisfy:

- Intrinsic properties we want it to have
  - e.g. to contract like a rubberband etc.
- Extrinsic image-related properties we want it to have
  - e.g. to get repelled by dark areas in the image, or by sharp edges

Snake Energy

Physical Analogy

- Snake properties can be thought of with physical analogies:
  - A snake that is meant to shrink around a shape is like a rubberband
  - A snake that is meant to approximate a shape from within can be thought as a balloon that inflates
  - Extrensic properties can also be thought of in this way: a grey-level gradient in the image can be thought as a hill to climb

Snake energy

- Physical deformable models can be explained in terms of energy
  - a rubberband has internal energy which it tries to minimize
- Snakes can be modelled in terms of energy as well
- A snake then seeks to minimise its energy
- It deforms to fit a local minimum

Snake Energy

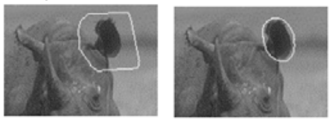
- Snake energy is made up of internal and external parts added together:

$$E = E_{internal} + E_{external}$$

- We define the energy of a snake so that states we want it to be in have low energy
  - The goal of the snake is then to minimize its energy
  - It deforms until it reaches a local minimum

Shrinking around a Dark Area

Example



- To make a snake shrink, we define internal energy to increase with the length of the snake, e.g.:

$$E_{internal} = A \sum_{\text{control points}} (\text{distance between control point and neighbour to left})^2$$

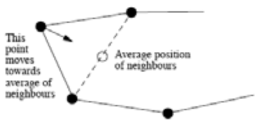
- To make it avoid dark area, we define external energy inversely proportional to grey-levels under the snake, e.g.:

$$E_{external} = -B \sum_{\text{control points}} I(\text{location of control point})$$

Shrinking around a Dark Area

Internal Property

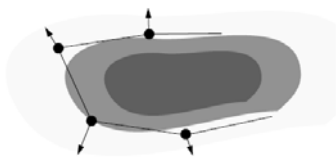
- Given our definition, the snake should try to reduce the distance between neighbouring control points in order to reduce its energy
- This can be implemented by moving a control point closer to the average of its neighbours (by a certain proportion)



Shrinking around a Dark Area

External Property

- Given our definition for external energy, the snake should try to move its control points to positions of increase intensity (i.e. lighter grey-level)
- This can be implemented by moving a control point in the direction of the gradient of image intensity



Shrinking around a Dark Area

- In each iteration, new positions are calculated for all control points, based on a combination of internal and external force.
- The x coordinate of a control point can be updated to:

$$x_{new} = x + \alpha\left(\frac{1}{2}(x_{left} + x_{right}) - x\right) + \beta[I(x+1, y) - I(x-1, y)]$$

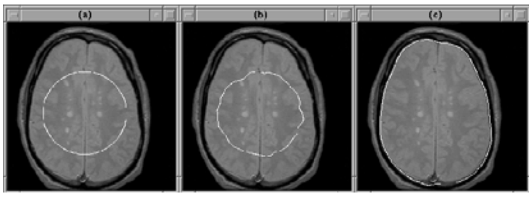
- The formula for updating the y coordinate is similar.
- The snake iteratively deforms until the elastic inward force is balanced by the outward force from the grey-level gradient

Snake Energy

Snake energy can be defined for many different properties

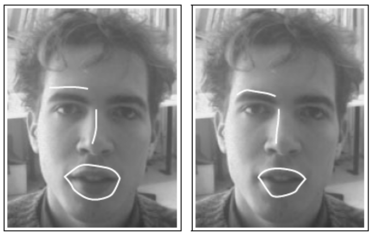
- Continuity Energy (smoothing)
  - influences the shape of the contour
  - setting this parameter too high results in a final contour that is very smooth but does not track image detail.
  - a setting that is too low allows the contour to move without restraint resulting in discontinuities
- Balloon energy
  - makes a snake inflate, e.g. to fill a concavity
- Attraction to edge features
  - define energy with respect to gradient

Balloon Energy



Snake Energy

What forces at work?



Initial configuration   Final configuration

Attraction to Edge Features

Here the external energy is defined so that the snake wants to lie on regions of high grey-level gradient.

The gradients are shown at the top; but they never actually have to be computed for the whole image.

The snake contracts to lie on the boundaries round the butterfly except where the elastic energy pulls it tight) even though at some points no clear boundary is defined in the image.

Snake Summary

Snakes are energy minimising splines

- Choose the properties — internal and external — that you want your contour to end up with.
- Express these as energy formulae and differentiate to get rules to move the control points.
- Iterate the rules so that the snake evolves to a good state.

Some further points

- Snakes are local, not global, so initial location must be provided
  - By hand: they were originally conceived as “power assist” for human operators
  - Randomly all over, or on the basis of some pre-processing
- Snakes are computationally very cheap
  - Compare with filter operations for edge detection!

Overview

- What is Computer Vision?
- Exercise: Image Structure for 3D
- Image Filtering, Convolution
- Scale Space
- Edge Detection
- The Hough Transform
- Active Contours

- Quick recap of the main concepts to wrap-up

Convolution

Linear filtering with masks

- Smoothing, differencing, combinations of both

- What does this mask do?

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

- And this one?

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

Convolution

Gradient Filters

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	8	8	8	8	8
0	0	0	8	8	8	8	8
0	0	0	8	8	8	8	8
0	0	0	8	8	8	8	8
0	0	0	8	8	8	8	8

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

Horizontal gradient

0	0	0	0	0	0
0	0	0	0	0	0
0	3	3	0	0	0
0	4	4	0	0	0
0	4	4	0	0	0
0	4	4	0	0	0

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

Vertical gradient

0	0	0	0	0	0
0	0	3	4	4	4
0	0	3	4	4	4
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Convolution

Sobel Operator

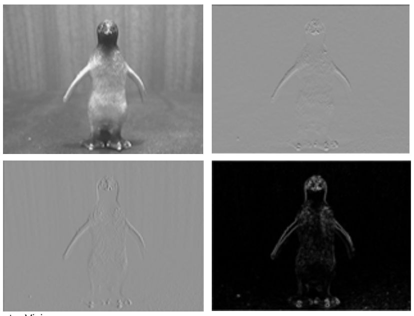
- Based on the 3x3 horizontal gradient (G_x) and vertical gradient masks (G_y) applied to image A

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

- At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$\mathbf{G} = \sqrt{\mathbf{G_x}^2 + \mathbf{G_y}^2}$$

Convolution: Sobel Operator



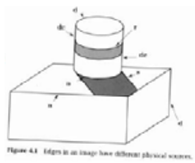
Edge Detection

Looked at ways of finding edges.

But how can we interpret them?

Edges can be …

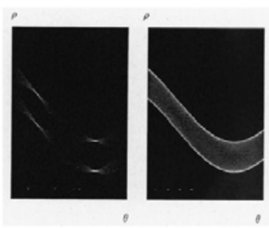
- object-background boundaries (edges [d] and depth discontinuity [dc])
- object-object boundaries (those correspond not necessarily to object boundaries [n])
- shadows [s]
- discontinuities of object texture [r]
- discontinuities of surface normals [n]



Hough Transform

Finding straight lines in an image

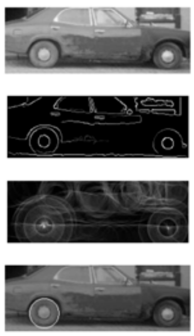
- Hough Transform for a square (left) and a circle (right)



Hough Transform

Can also find other shapes

- e.g. circles



Grey-level image

Canny edges (Edge orientation ignored.)

Accumulator array — the transform itself

Most prominent circle

Snakes

Fitting of contours to structure in an image

- Based on physical model
- Computationally cheap
  - very local: computing movement of control points
  - in contrast to filters and transforms
  - good for interactive tasks: human operator approximates the shape, snake then fits itself to detail

Computer Vision

This lecture has told you a little about

- Some algorithms and representations, e.g.
  - convolution, grey level arrays and scale space
  - Hough transform, straight lines and Hough spaces
  - energy minimisation and active contours

But there is of course a lot more …

- Stereoscopic vision …
- Active vision …
- Dynamic vision …
- Many specific algorithms that are important
  - e.g. ellipse-fitting, shape from texture, …