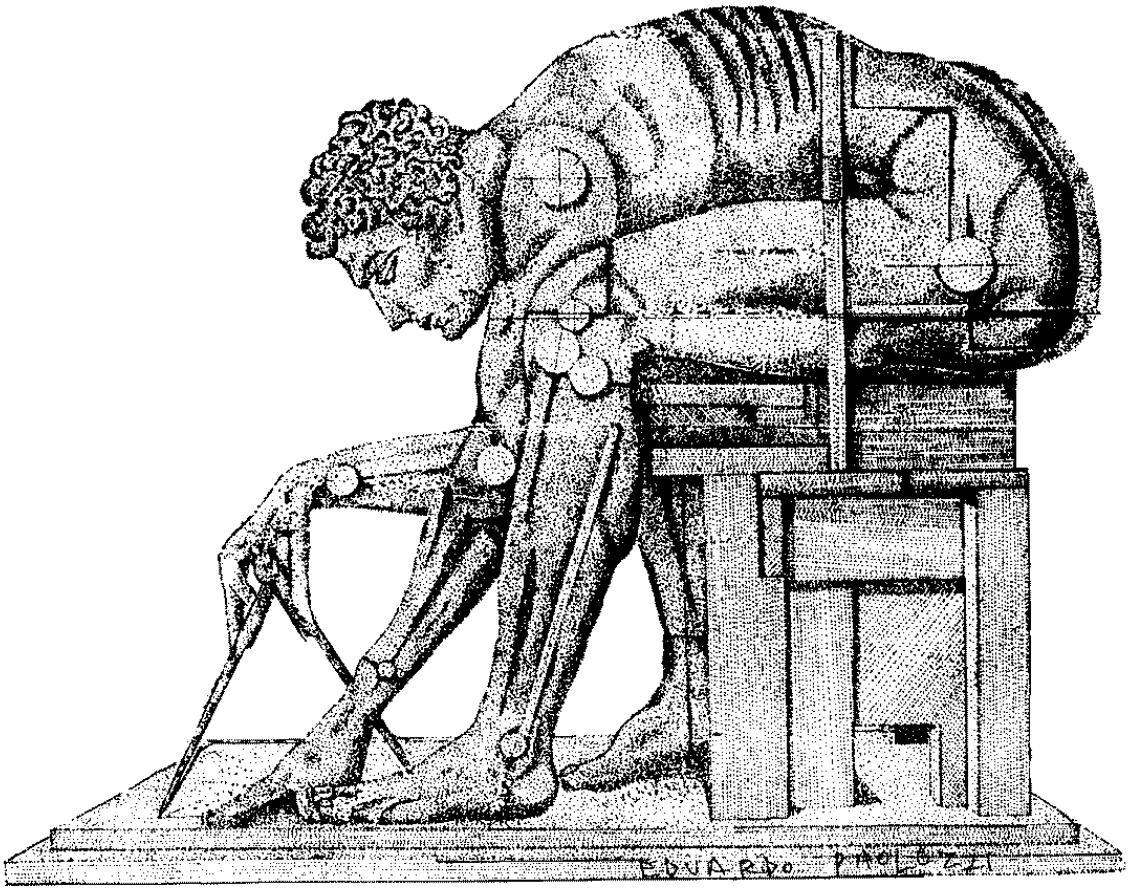
# University of Cambridge
# Engineering Part IIB

# Module 4F12: Computer Vision and Robotics

# Handout 1: Introduction



Roberto Cipolla and Richard Turner
October 2014

# What is computer vision?

**Vision** is about discovering from images what is present in the scene and where it is. It is our most powerful sense.

In **computer vision** a camera (or several cameras) is linked to a computer. The computer automatically interprets images of a real scene to obtain useful information (3**R's**: registration, recognition and reconstruction) and then acts on that information (e.g. for navigation, manipulation or recognition).

**images** $\rightarrow$ **representation**
**perception** $\rightarrow$ **actions**

It is *not*:

**Image processing:** image enhancement, image restoration, image compression. Take an image and process it to produce a new image which is, in some way, more desirable.

**Pattern recognition:** classifies patterns into one of a finite set of prototypes.

# Why study computer vision?

1. Intellectual curiosity — how do *we* see?
2. Replicate human vision to allow a machine to see — many industrial applications.

# Applications

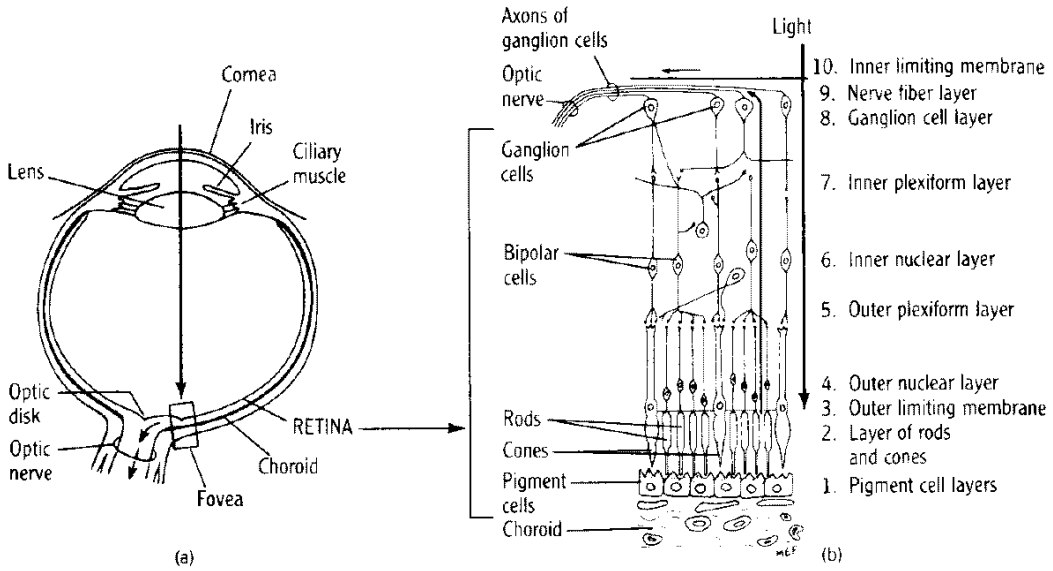
- Automation of industrial processes
  - Object recognition.
  - Visual inspection.
  - Robot hand-eye coordination
  - Robot navigation.

# Applications

- Space and Military
  - Remote sensing
  - Surveillance - target detection and tracking
  - UAV localisation
- Surveillance and tracking (traffic, aircraft, watching humans and *motion capture*)
- Human-computer interaction
  - Face detection and recognition.
  - Gesture-based HCI (e.g. new interfaces for games consoles)
  - Image search and retrieval from video and image databases
  - Mobile-phone applications - target/object recognition, mosaicing
  - Augmented reality
- 3D modelling, measurement and visualisation
  - 3D model building from image sequences
  - Photogrammetry.
- Automotive applications and autonomous vehicles

# How to study vision? The eye

Let's start with the human visual system.




- Retina measures about 1000 mm$^2$ and contains about $10^8$ sampling elements (rods) (and about $10^6$ cones for sampling colour).
- The eye's spatial resolution is about 0.01° over a 150° field of view (not evenly spaced, there is a fovea and a peripheral region).
- Intensity resolution is about 11 bits/element, spectral resolution is about 2 bits/element (400–700 nm).
- Temporal resolution is about 100 ms (10 Hz).
- Two eyes (each about 2cm in diameter), separated by about 6cm.

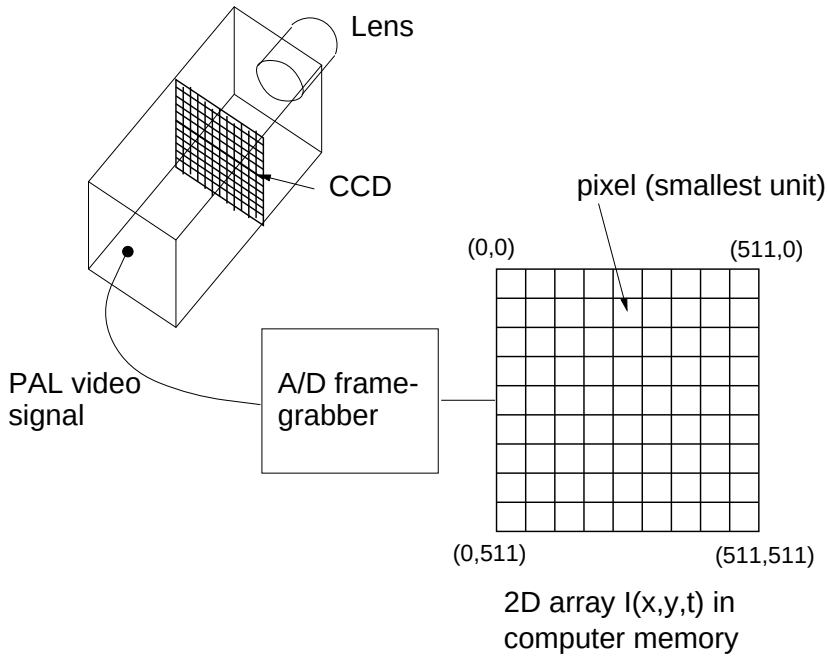
- A large chunk of our brain is dedicated to processing the signals from our eyes - a data rate of about 3 GBytes/s!

# Why not copy the biology?

- There is no point copying the eye and brain — human vision involves 60 billion neurons!
- Evolution took its course under a set of constraints that are very different from today's technological barriers.
- The computers we have available cannot perform like the human brain.
- We need to understand the underlying principles rather than the particular implementation.

Compare with flight. Attempts to duplicate the flight of birds failed!

# The camera




- A typical digital SLR CCD measures about $24 \times 16$ mm and contains about $6 \times 10^6$ sampling elements (pixels).
- Intensity resolution is about 8 bits/pixel for each colour channel (RGB).
- Most computer vision applications work with monochrome images.
- Temporal resolution is about 40 ms (25 Hz)
- One camera gives a raw data rate of about 400 MBytes/s.

The CCD camera is an adequate sensor for computer vision.

# Image formation

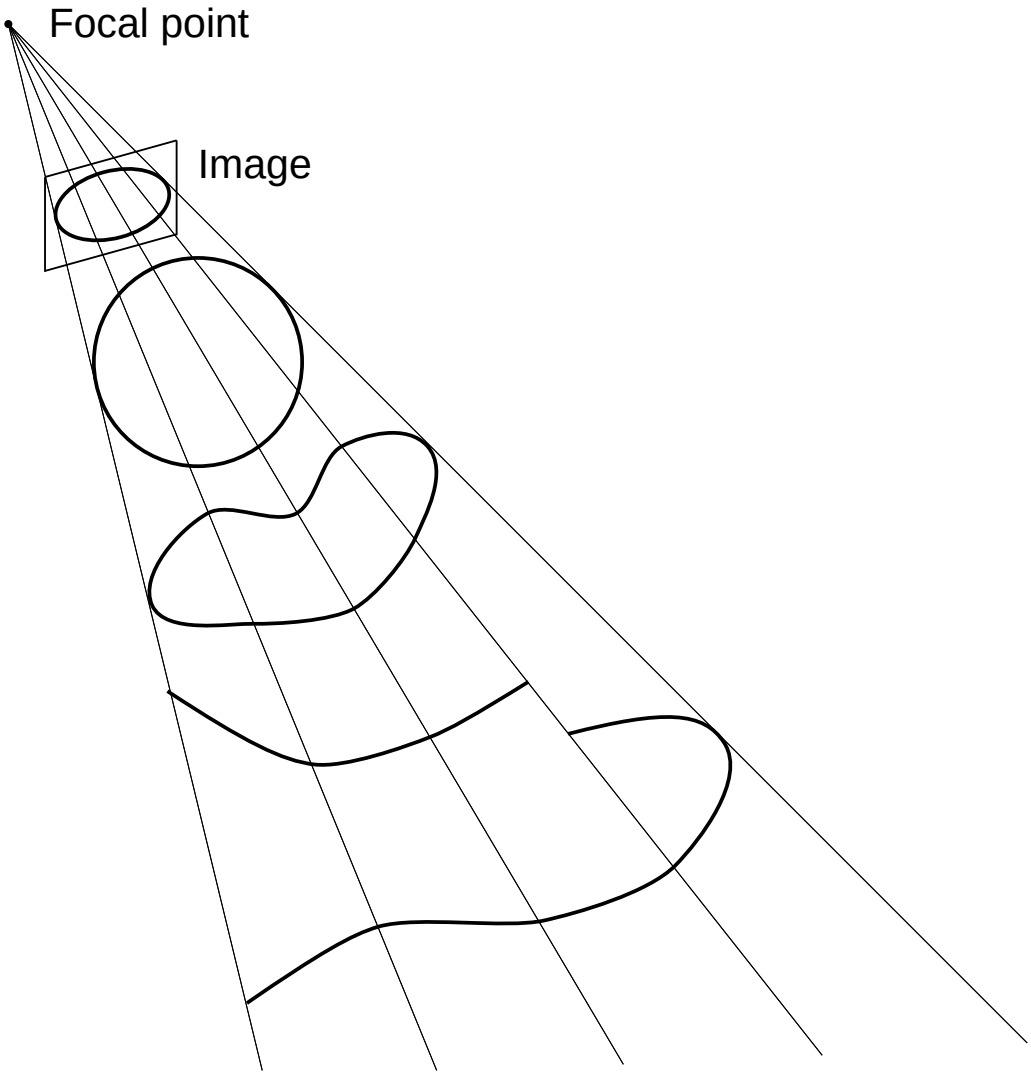
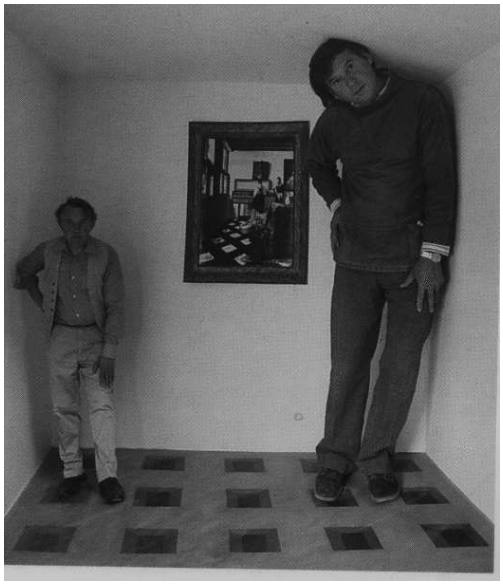
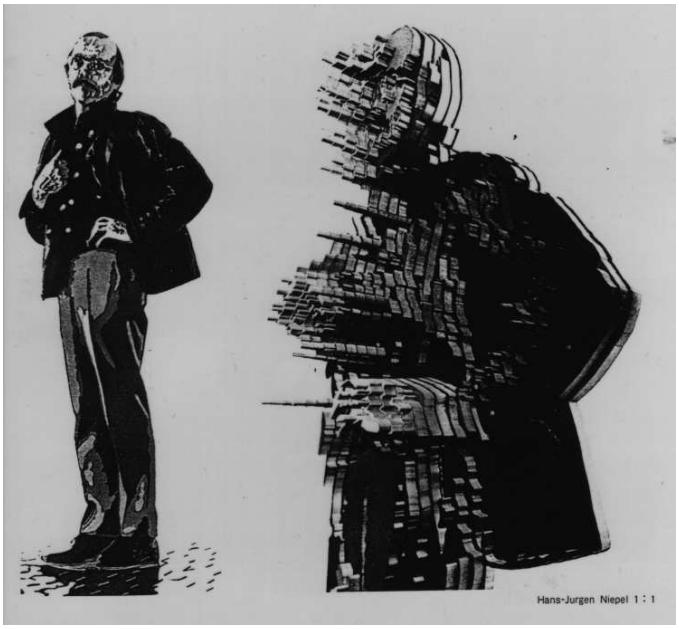



Image formation is a many-to-one mapping. The image encodes nothing about the *depth* of the objects in the scene. It only tells us along *which* ray a feature lies, not *how far* along the ray. The inverse imaging problem (inferring the scene from a single image) has no unique solution.

# Ambiguities in the imaging process



Two examples showing that image formation is a many-to-one mapping. The Ames room and two images of the same 3D structure.



# Vision as information processing

David Marr, one of the pioneers of computer vision, said:
" *One cannot understand what seeing is and how it works unless one understands the underlying information processing tasks being solved.*"

From an information processing point of view, vision involves a huge amount of data reduction:

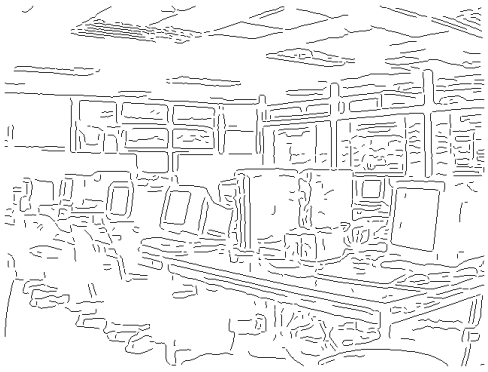
| | | |
|---|---|---|
| images | $\rightarrow$ | generic salient features |
| 10 MBytes/s (mono CCD) | | 10 KBytes/s |
| salient features | $\rightarrow$ | representations and actions |
| 10 KBytes/s | | 1–10 bits/s |

Vision is also about resolving the ambiguities inherent in the imaging process, by drawing on a set of constraints (AI). But where do the constraints come from? We have two options:

1. Use more than one image of the scene.
2. Make assumptions about the world in the scene.

# Feature extraction

The first stages of most computer vision algorithms perform feature extraction. The aim is to reduce the data content of the images while preserving the useful information they contain.

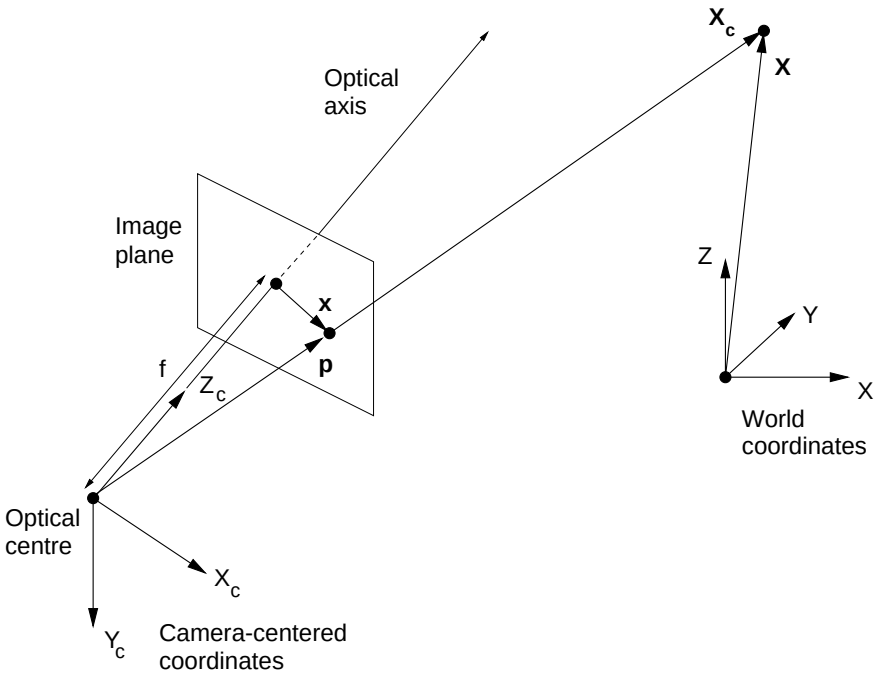


The most commonly used features are edges, which are detected along 1-dimensional intensity discontinuities in the image. Automatic edge detection algorithms produce something resembling a line drawing of the scene.

Corner detection is also common. Corner features are particularly useful for motion analysis.

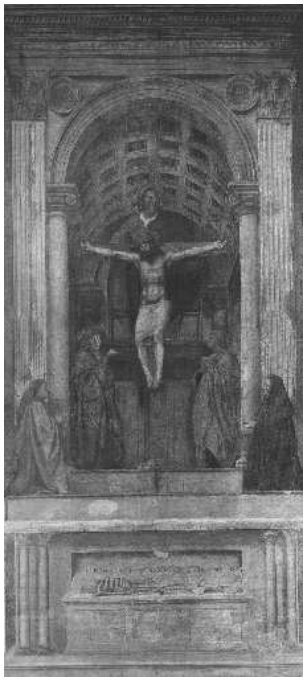
# Camera models

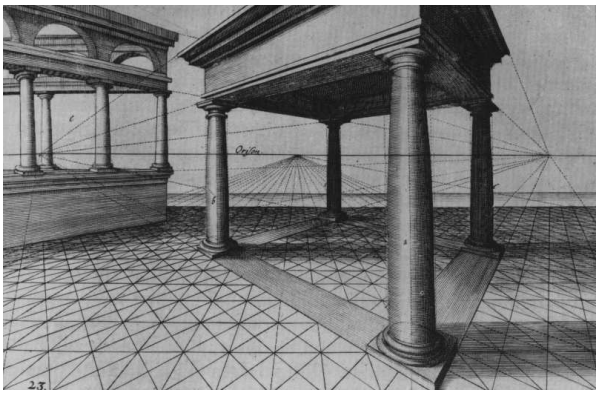
Before we attempt to interpret the image (or the features extracted from the image), we have to understand how the image was formed. In other words, we have to develop a **camera model**.



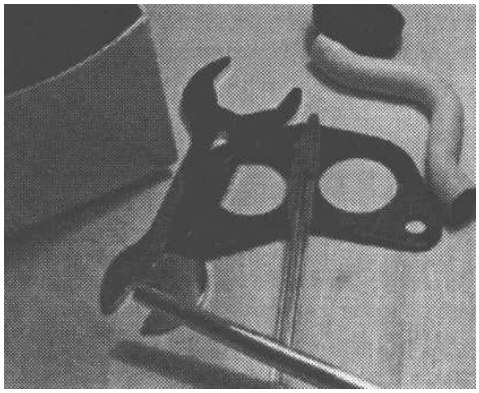

Camera models must account for the position of the camera, perspective projection and CCD imaging. These geometric transformations have been well-understood since the C14th. They are best described within the framework of **projective geometry**.

# Camera models

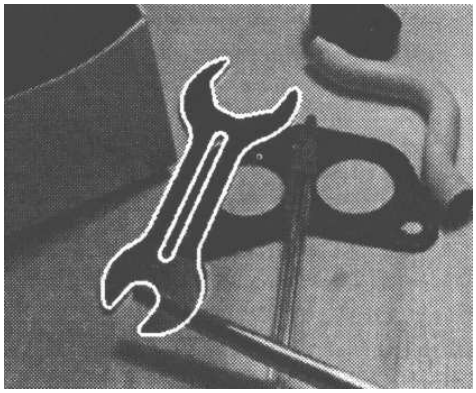




Having established a camera model, we can predict how known objects will appear in an image, and can attempt **object recognition**.



Cluttered scene

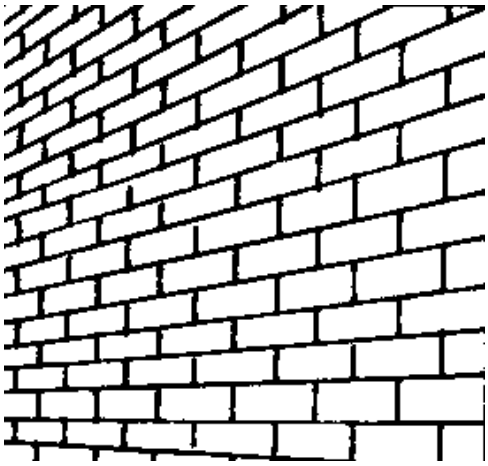


Spanner identified

# Shape from texture

Texture provides a very strong cue for inferring surface orientation in a single image. It is necessary to assume *homogeneous* or *isotropic* texture. Then, it is possible to infer the orientation of surfaces by analysing how the texture statistics vary over the image.

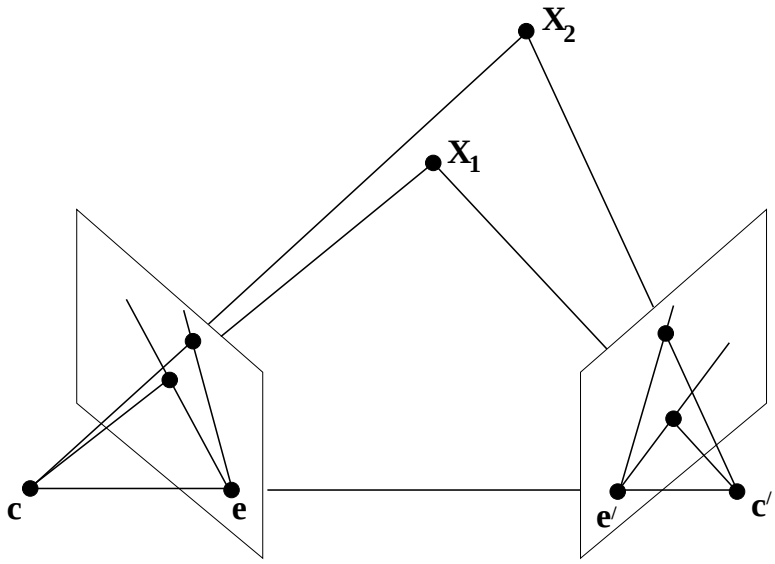
Here we perceive a vertical wall slanted away from the camera.



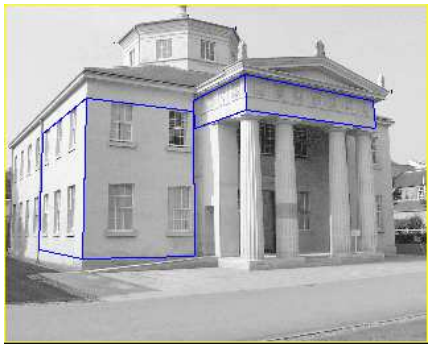
And here we perceive a horizontal surface below the camera.

# Stereo vision

Having two cameras allows us to triangulate on features in the left and right images to obtain depth. It is even possible to infer useful information about the scene when the cameras are not **calibrated**.




Stereo vision requires that features in the left and right image be matched. This is the well-known and difficult **correspondence problem**.

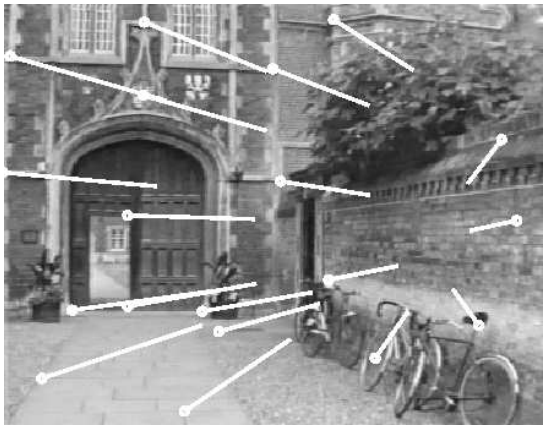


# Structure from motion

Related to stereo vision is a technique known as **structure from motion**. Instead of collecting two images simultaneously, we allow a single camera to move and collect a sequence of images.

As the camera moves, the motion of some features (in this case corner features) is **tracked**.

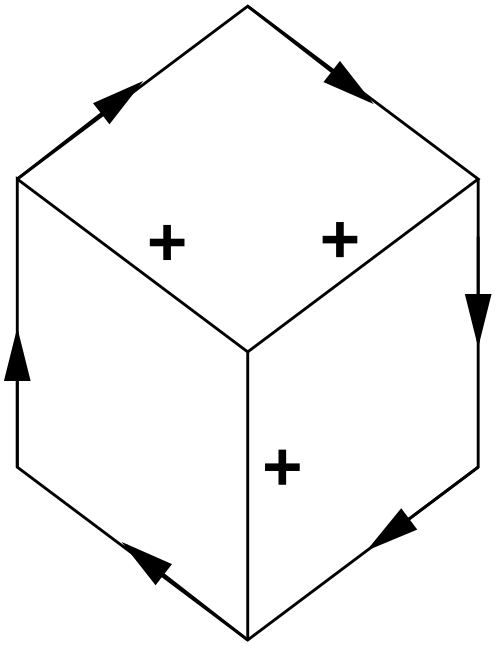
The trajectories allow us to say something about the structure in the scene and the motion of the camera.



Structure from motion algorithms are sensitive to independently moving objects in the scene.

# Shape from line drawings

It is possible to infer scene structure from line drawings under certain assumptions. If we assume we are looking at trihedral-vertex polyhedra, we can propagate constraints at each vertex to infer the shape (only 16 labels for a trihedral junction are visibly feasible).



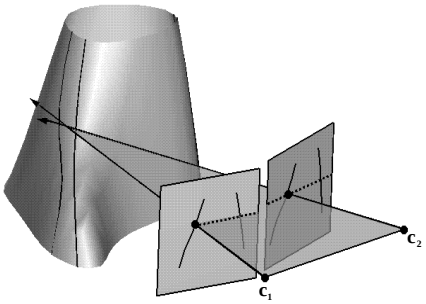

Matisse

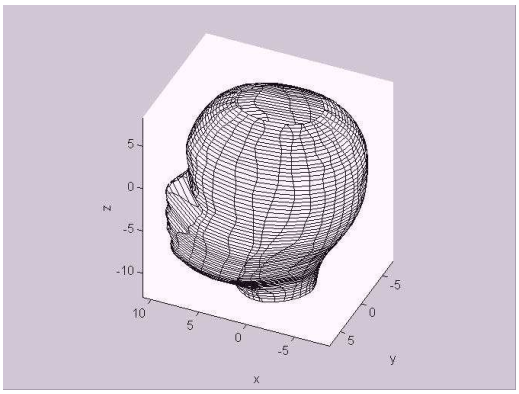
We can also interpret line drawings of curved surfaces. Under assumptions of smoothness, the **apparent contours** (profile) and **cusps** in the Matisse tell us something about the curvature of the surfaces. A full interpretation of this image, however, requires some top-down knowledge.

# Shape from contour

A curved surface is bounded by its **apparent contours** in an image. Each contour defines a set of **tangent planes** from the camera to the surface.

As the camera moves, the contour generators "slip" over the curved surface. By analysing the deformation of the apparent contours in the image, it is possible to reconstruct the 3D shape of the curved surface.

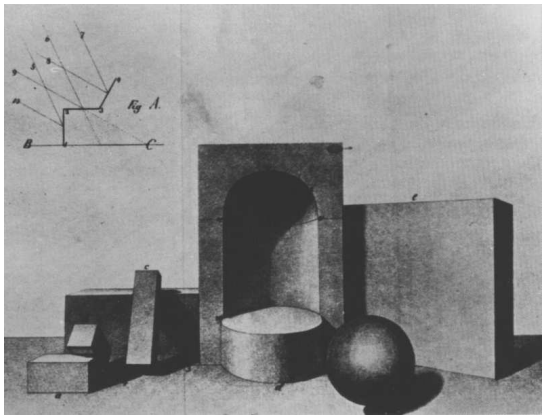


# Shape from shading

It is possible to infer the structure in a scene from the **shading** observed in the image.

Assumptions we make include a Lambertian light source, isotropic surface reflectance, and a top-lit scene.



Peter Paul Rubens
*Samson and Delilah* (detail)

The **self shadows** of an object give particularly rich information.

# Human visual capabilities

Our visual system allows us to successfully interpret images under a wide range of conditions. It is not surprising that we can cope with normal stereo vision: we have two eyes to use for triangulation, and a number of other cues (motion, shading etc.) to help us.

More surprising is our ability to interpret a wide range of images with limited cues.

# Geometrical framework

The first part of the course will focus on generic computer vision techniques which make minimal assumptions about the outside world. This means we'll be concentrating on the theory of perspective, stereo vision and structure from motion.

We typically use a geometric framework:

1. Reduce the information content of the images to a manageable size by extracting salient features, typically **edges** or **blobs** . (These features are generic and substantially *invariant* to a variety of lighting conditions.)
2. Model the imaging process, usually as a perspective projection and express using projective transformations.
3. Invert the transformation using as many images and constraints as necessary to extract 3D structure and motion.

# Statistical framework

Geometry alone is only a part of the solution. In the second part of the course we will introduce techniques which learn from the visual world. They are part of a statistical framework to understanding vision and for building systems which:

1. Have the ability to test hypotheses
2. Deal with the ambiguity of the visual world
3. Are able to fuse information
4. Have the ability to learn

Many of these requirements can be addressed by reasoning with probabilities and are the subject of other advanced courses on Statistical Pattern Recognition and Machine Learning.

# Syllabus

### 1. Introduction

- Computer vision: what is it, why study it and how?
- Vision as an information processing task
- A geometrical framework for vision
- 3D interpretation of 2D images

### 2. Image structure

- Image intensities and structure: edges, corners and blobs
- Edge detection, the aperture problem, corner detection, blob detection
- Texture.
- Feature descriptors and matching of features.

### 3. Projection

- Orthographic projection
- Planar perspective projection, vanishing points and lines.
- Projection matrix, homogeneous coordinates
- Camera calibration, recovery of world position
- Weak perspective, the affine camera

### 4. Stereo vision and Structure from Motion

- Epipolar geometry and the essential matrix
- Recovery of depth
- Uncalibrated cameras and the fundamental matrix
- The correspondence problem
- Structure from motionm
- 3D shape with multiple view stereo.

### 5. Object detection and and recognition

- Basic target detection and tracking
- Template matching. Chamfer matching and template trees
- Learning to recognise objects
- Random decision forests, support vector machines and boosting
- Deep learning with convolutional neural networks.

**Course book:** V. S. Nalwa. *A Guided Tour Of Computer Vision*, Addison-Wesley, 1993 (CUED shelf mark: NO 219).

# Further reading

Students looking for a deeper understanding of computer vision might wish to consult the following publications, many of which are available in the CUED library.

**Journals**
International Journal of Computer Vision
IEEE Transactions on Pattern Analysis and Machine Intelligence
Image and Vision Computing
Computer Vision, Graphics and Image Processing

**Conference proceedings**
International Conference on Computer Vision
European Conference on Computer Vision
Computer Vision and Pattern Recognition Conference
British Machine Vision Conference

**Books**
R. Cipolla and P. Giblin *Visual Motion of Curves and Surfaces.* CUP, 1999.

D.A. Forsyth and J. Ponce. *Computer Vision - A Modern Approach.* Prentice Hall 2003.

* R. Hartley and A. Zisserman. *Multiple View Geometry.*CUP 2000.

J. J. Koenderink. *Solid shape.* MIT Press, 1990.

D. Marr. *Vision: a computational investigation into the human representation and processing of visual information.* Freeman, 1982.

* S.J.D. Prince *Computer Vision: Models, Learning and Inference.* CUP, 2012.

* R. Szeliski. *Computer Vision: algorithms and applications.* Springer, 2011.

B. A. Wandell. *Foundations of vision.* Sinauer Associates, 1995.

**See also the bibliographies at the end of each handout.**

# Mathematical Preliminaries

## Linear least squares

Consider a set of $m$ linear equations

$$\mathrm{A}\mathbf{x} = \mathbf{b}$$

where $\mathbf{x}$ is an $n$-element vector of unknowns, $\mathbf{b}$ is an $m$-element vector and A is an $m \times n$ matrix of coefficients. If $m > n$ then the set of equations is *over-constrained* and it is generally not possible to find a precise solution $\mathbf{x}$.

The equations can, however, be solved in a **least squares** sense. That is, we can find a vector $\mathbf{x}$ which minimizes

$$\sum_{i=1}^{m} r_i^2$$

where

$$\mathrm{A}\mathbf{x} = \mathbf{b} + \mathbf{r}$$

$\mathbf{r}$ is the vector of *residuals*.

The least squares solution is found with the aid of the **pseudo-inverse**:

$$\mathrm{A}^\dagger = \left(\mathrm{A}^T\mathrm{A}\right)^{-1}\mathrm{A}^T$$

The least squares solution is then given by $\mathbf{x} = \mathrm{A}^\dagger\,\mathbf{b}$.

# Mathematical Preliminaries

## Eigenvectors and eigenvalues

Often the equations can be written as a set of $m$ linear equations

$$\mathrm{A}\mathbf{x} = \mathbf{0}$$

where $\mathbf{x}$ is an $n$-element vector of unknowns and A is an $m \times n$ matrix of coefficients.

A non-trivial solution for $\mathbf{x}$ (up to an arbitrary magnitute) can be found if $m > n$. The solution is chosen to minimize the residuals given by $|\mathrm{A}\mathbf{x}|$ subject to $|\mathbf{x}| = 1$.

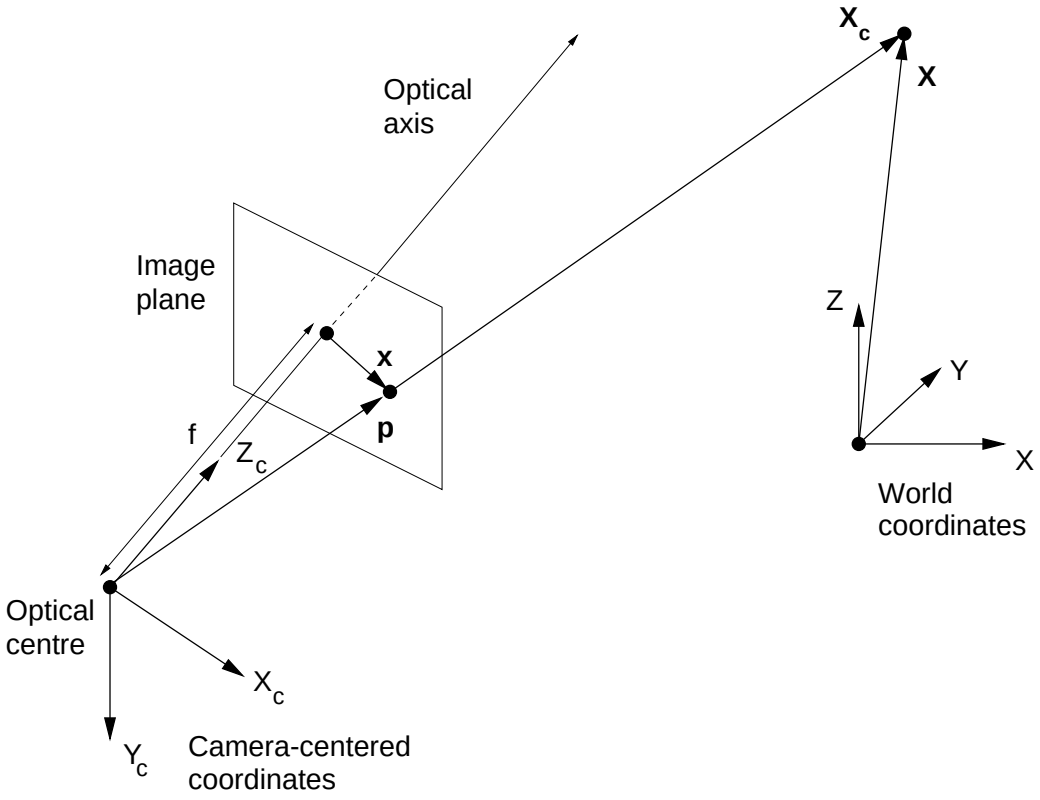
By considering Rayleigh's Quotient:

$$\lambda_1 \leq \frac{\mathbf{x}^T \mathrm{A}^T \mathrm{A}\mathbf{x}}{\mathbf{x}^T \mathbf{x}} \leq \lambda_n$$

it is easy to show that the solution is the eigenvector corresponding to the smallest eigenvalue of the $n \times n$ symmetric matrix $\mathrm{A}^T\mathrm{A}$.

# Notation

## Metric coordinates




### World coordinates

| | |
|---|---|
| $\mathbf{X} = (X, Y, Z)$ | Point in 3D space |
| $\mathbf{X}^p = (X, Y)$ | Point on 2D plane |
| $\mathbf{X}^l = (X)$ | Point on 1D line |

### Camera-centered coordinates

| | |
|---|---|
| $\mathbf{X}_\mathrm{c} = (X_c, Y_c, Z_c)$ | Point in 3D space |
| $\mathbf{p} = (x, y, f)$ | Ray to point on image plane |
| $\mathbf{x} = (x, y)$ | Image plane coordinates |

### Pixel coordinates

| | |
|---|---|
| $\mathbf{w} = (u, v)$ | Pixel coordinates |

# Notation

## Projection and transformation matrices

| | |
|---|---|
| $\mathrm{R}$ | Rotation matrix (orthonormal) |
| $\mathbf{T}$ | Translation vector (3 element) |
| $\mathrm{P}_r$ | Rigid body transformation matrix (3D) |
| $\mathrm{P}_p$ | Perspective projection matrix |
| $\mathrm{P}_{pll}$ | Parallel projection matrix (weak perspective) |
| $\mathrm{P}_c$ | CCD calibration matrix |
| $\mathrm{P}_{ps}$ | Overall perspective camera matrix |
| $\mathrm{P}_{wp}$ | Overall weak perspective camera matrix |
| $\mathrm{P}$ | Overall projective camera matrix |
| $\mathrm{P}_{aff}$ | Overall affine camera matrix |
| $[\,]^p$ | Superscript for plane imaging matrices |
| $[\,]^l$ | Superscript for line imaging matrices |

## Stereo

| | |
|---|---|
| $\mathbf{X}_\mathrm{c}$, $\mathbf{p}$, $\mathbf{w}\ldots$ | Left camera quantities |
| $\mathbf{X}'_\mathrm{c}$, $\mathbf{p}'$, $\mathbf{w}'\ldots$ | Right camera quantities |
| $\mathbf{p}_\mathrm{e}$, $\mathbf{p}'_\mathrm{e}$ | Rays to epipoles |
| $\mathbf{w}_\mathrm{e}$, $\mathbf{w}'_\mathrm{e}$ | Pixel coordinates of epipoles |
| $\mathrm{E}$ | Essential matrix |
| $\mathrm{F}$ | Fundamental matrix |

## Motion

| | |
|---|---|
| $\mathbf{v} = (\dot{x}, \dot{y})$ | Image motion field |
| $\mathbf{U}$ | Camera's linear velocity |
| $\Omega$ | Camera's angular velocity |
| $\Delta$ | Motion parallax vector |