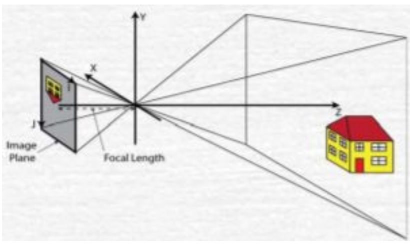
Introduction

- Computer Vision is about understanding images
    - Greyscale, Colour, Multi-Spectral
    - Snapshots or Video
    - Taken by static or moving cameras
    - Taken of a static or moving scene
    - Taken with a calibrated or un-calibrated camera
- Can drive:
    - **Inspection** - e.g. industrial inspection
    - **Analysis** - e.g. Surveillance/Forensics
    - **Control** - e.g. biometrics, landmine detection, medical imaging
- *"What we experience, apparently directly, is actually very different from what is recorded by our sense organs"*
- Applications: AR, Gaming, AutoDriver, Robot Vision

Learning Goals

- **Understand**
    - **the broad subject of computer vision**
    - **the main algorithmic processes used to manipulate images**
    - **How information may be extracted from images, and the associated problems**
- **Be able to**
    - **Describe the various operations both algorithmically and mathematically**
    - **Code and test basic vision algorithms**
    - **Develop potential solutions to complex vision problems**

Images

- Camera Models
    - Components: a photosensitive image plane, housing and lenses
    - Mathematical model needed - simple pinhole model, distortions need to be considered
        - 

        - 3D point translated to a 2D image (x,y,z) -> (i,j)
        - $$\begin{bmatrix} i.w \\ j.w \\ w \end{bmatrix} = \begin{bmatrix} f_i & 0 & c_i \\ 0 & f_j & c_j \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$
        - W is taken as the Scaling factor in the homogeneous coordinates being used to describe the image points
        - F_i and f_j describe the combination of camera focal length and the size of the pixels in the I and J directions respectively

        - (c_i,c_j) are the coordinates of the point at which the optical axis intersects the image plane (**optical centre**)
            - Note: this is the perpendicular line from the image plane which extends through the pinhole of the camera
- Digital Images
    - Images are (generally) a 2D projection of a 3D scene
    - Continuous 2D function
    - To process with a computer we need discrete representation
        - Conversion into an MxN matrix
        - Where each element is given an integer value - i.e. the continuous range (colour spectrum) is split into k intervals (where k is typically 256)
        - **Sampling**
            - Digital images are created with 2D arrays of of photosensitive elements with small borders between them - can result in data loss
            - Big issue is that pixels represent an average value (luminance and chrominance) over a discrete area which in the real world could source from a single object, but also multiple objects
        - **Quantisation**
            - Each pixel in a digital image f(i,j) is a function of scene brightness, the brightness values are continuous but need to be quantised
            - Typically the number of brightness levels per channel is $k=2^b$ where b is the number of bits (typically 8)
            - How many bits to use? The more you use the more memory occupied, the less you use the less information in the image
        - **We want enough samples to not waste space and time, and get exactly enough information**
- **Colour Images**
    - **Luminance only** - simple representation, humans can understand
    - **Colour images** (luminance + chrominance)
        - multiple channels (typically 3)
        - ~16.8 million colours - more complex to process
        - Facilitates more operations
        - **RGB images**
            - Red (700nm), Green (546nm), Blue (436nm)
            - Colours are combined on viewing
            - RGB -> GreyScale
                - **Y = 0.299R + 0.587G + 0.114B**
            - Camera photosensitive elements
                - Separate for red green blue - sometimes sensitive to all

        - Bayer pattern
    - **CMY Images**
      - Cyan-Magenta-Yellow
      - Secondary colours, subtractive scheme
        - **C = 255 - R**
        - **M = 255 - G**
        - **Y = 255 - B**
    - **YUV**
      - Previously used for analogue tv signals
      - Conversion from RGB
        - **Y = 0.299R + 0.587G + 0.114B**
        - **U = 0.492 * (B-Y)**
        - **V = 0.877 * (R-Y)**
    - **HLS**
      - Hue, luminance, saturation
      - Separation of luminance and chrominance
        - Hue = 0...360 (circular)
        - Luminance = 0...1
        - Saturation = 0...1
      - $$V_{max} \leftarrow max(R,G,B)$$ $$V_{min} \leftarrow min(R,G,B)$$ $$L \leftarrow \frac{V_{max}+V_{min}}{2}$$ $$S \leftarrow \begin{cases} \frac{V_{max}-V_{min}}{V_{max}+V_{min}} & \text{if } L < 0.5 \\ \frac{V_{max}-V_{min}}{2-(V_{max}+V_{min})} & \text{if } L \geq 0.5 \end{cases}$$ $$H \leftarrow \begin{cases} 60(G-B)/(V_{max}-V_{min}) & \text{if } V_{max}=R \\ 120+60(B-R)/(V_{max}-V_{min}) & \text{if } V_{max}=G \\ 240+60(R-G)/(V_{max}-V_{min}) & \text{if } V_{max}=B \end{cases}$$
- **Noise**
  - Affects most images, degrades them, interferes with processing
  - Measuring noise
    - Signal to Noise ratio
      - $$S/N\ ratio = \frac{\sum_{(i,j)} f2(i,j)}{\sum_{(i,j)} v2(i,j)}$$
    - Types of noise:
    - Gaussian
      - Good approximation to real noise - distribution is Gaussian (has mean and std. dev.)
    - Salt and Pepper
      - Impulse noise - noise is maximum or minimum values
- **Smoothing**
  - Removing or reducing noise

- Linear smoothing transformations
    - Image averaging - average on n images (***assumes static camera & scene, statistical independence***)
    - Local averaging and Gaussian smoothing
        - Filtering / Convolution
        - Linear transformation
            - Convolution mask
        - Non-linear: some logical operation performed on a local region
        - Averaging filters
            - Local neighbourhood
            - Different masks available: local average, gaussian

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

            - $f(i,j) = \sum\sum_{(m,n)\in O} h(i-m, j-n).g(m,n)$ $\quad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
            -
        - How to determine acceptable results: Size of noise, blurring of edges
    - **Issues with smoothing: blurring sharp edges, quality degradation?**
- Non-Linear Transformations
    - Rotating mask
        - Define a number of masks/regions
            - Mask size and shape
            - Use the average of one of the masks - which? **Most homogeneous**
        - Algorithm
            - For each point: calculate dispersions, assign output point average of mask with minimum dispersion
            - $$\sigma^2 = \frac{1}{n}\sum_{(i,j)\in R}\left[g(i,j) - \frac{1}{n}\sum_{(i',j')\in R} g(i',j')\right]^2$$
        - Iterative application: Convergence; Effects of mask size
        - Effects: **noise suppression and image sharpening**
    - Median filter
        - Use the median value, not affected by noise (ignores average)
        - **Doesnt blur edges and can be applied iteratively**
        - Damages thin lines and sharp corners (can change shape to mitigate this)
        - Computationally expensive
    - Bilateral filter

        - Weight local pixels
            - Distance from centre
            - Difference in colour/intensity space

$$f(i,j) = \frac{1}{W_p} \sum_{(m,n)\in\Omega} g(m,n) f_R(\|g(m,n) - f(i,j)\|) f_S\left(\sqrt[2]{(m-i)^2 + (n-j)^2}\right)$$

$$W_p = \sum_{(m,n)\in\Omega} f_R(\|g(m,n) - f(i,j)\|) f_S\left(\sqrt[2]{(m-i)^2 + (n-j)^2}\right)$$

        - Preserves edges
        - **Causes staircase effect, introduction of false edges**
- **Image Pyramids**
    - Process images at multiple scales efficiently
    - Technique
        - Smooth image (regularly gaussian)
        - Sub-sample (usually by a factor of 2)