COMP3005/Computer Vision

1

Building machines that see

Summary

A computer vision system is not just software: it is any **hardware**, **software** and **wetware** (i.e. humans) that make up the complete system. To engineer a computer vision system, you need to think about how you can achieve the required level of robustness by constraining the problem at hand and incorporating sufficient invariance to potentially changing environmental factors.

Key points

System Design

- All computer vision systems operate within an **environment**
  - The environment could be physical (a factory, the world, …), or purely virtual.
  - Environmental factors can affect the performance of the system.
- Computer vision systems need to be **robust**.
- A measure of robustness is **repeatability**.
- Software and hardware can be designed in to **invariant** to certain environmental factors.
  - Common example: invariance to rotation - the system should still work if the object being imaged appears at a different orientation in the image.
- An important part of designing a robust computer vision system is to choose **constraints** to apply to the environment and system.
  - Constraints restrict the need for certain types of invariance, making the system simpler (and potentially more cost effective).
  - Examples:
    - Control over lighting: diffuse versus directional; colour; wavelength (i.e. IR/UV); filters; etc.
    - Control over camera: type; frame-rate; filters; etc.
    - Enclosure to mitigate external environmental changes.
    - Limiting the allowed/possible orientations of the object being imaged relative to the camera (thus requiring less rotation invariance).

Colour-spaces

- Most digital cameras and display devices work in RGB (Red, Green, Blue) colour-space.
- Luminance, or brightness, can be defined as a (weighted) average of R, G and B.
- RGB colour-space is said to be "*coupled*": changing the amount of luminance changes all three values.
  - This can be a problem for software that needs to be invariant to illumination changes
  - By mapping RGB into a different colour-space we can decorrelate luminance
- HSV (Hue, Saturation, Value) is a commonly used colour-space in computer vision that decorrelates luminance.
  - The value component in the luminance
  - Saturation is a measure of how vivid the colour is
  - Hue is the raw colour (measured as an angle around a colour wheel)
    - Problem: It's cyclic, so red = 0° = 360°. We have to deal with this carefully in any software.

Further reading

- White papers on lighting for industrial vision: http://www.ni.com/white-paper/6901/en/
- Further information on HSV (and some alternative) colour-spaces, including code: http://archive.is/NEzmQ
- Mark's Book (third edition): Appendix 4 covers colour images and colour-spaces.

Practical exercises

- Get the slides/handouts/demos: http://github.com/jonhare/COMP3005
- Grab a copy of the OpenIMAJ tutorial:
  - PDF version here: http://www.openimaj.org/tutorial-pdf.pdf
  - Online HTML version here: http://www.openimaj.org/tutorial/
- Learn the basics of using OpenIMAJ to process images and video in Java
  - OpenIMAJ Tutorial chapters 1, 2, 7
- Explore different colour-spaces
  - Play with the `org.openimaj.image.colour.ColourSpace` class