Creative computing II: interactive multimedia

Volume 2: Perception, multimedia information retrieval and animation

C. S. Rhodes with S. Rauchas

2910**227**

2009

Undergraduate study in
Computing and Related Subjects

The material in this subject guide was prepared for the University of London External System by:

Christophe Rhodes, Department of Computing, Goldsmiths, University of London
Sarah Rauchas, Department of Computing, Goldsmiths, University of London.

The guide was produced by Sarah Rauchas and Christophe Rhodes, Department of Computing, Goldsmiths, University of London.

This is one of a series of subject guides published by the University.

This subject guide is for the use of University of London External System students registered for programmes in the field of Computing. The programmes currently available in these subject areas are:

BSc(Honours) in Computing and Information Systems
BSc(Honours) in Creative Computing
Diploma in Computing and Information Systems
Diploma in Creative Computing

Published 2009

Publisher:
The External System
Publications Office
University of London
Stewart House
32 Russell Square
London
WC1B 5DN

www.londonexternal.ac.uk

Contents

Preface

The first volume of the subject guide for **Creative Computing 2** focused on signals, signal processing, systems and their creative applications. This second volume complements the first by providing details about how the information-bearing signals of light and sound are perceived, both at a low level (providing details of the fundamental processes in the eye and the ear) and at a higher, more cognitive level. Knowing about these details of the processes of perception can motivate particular choices of signal processing techniques, and the construction of systems to exploit particular effects.

In addition, an understanding of the details of perception and cognitive processes is important for your creative practice, in that it will guide your production. Examples such as: choosing colour schemes that are distinct even to colour-blind observers; being able to estimate how to produce equally-loud sounds at different fundamental frequencies; understanding how to generate and display flicker-free animations – all of these depend on the range of human perceptive processes.

This subject guide is not a complete unit text on its own. It introduces topics and concepts, and provides some material to help you study the topics in the unit. Further reading is very important as you are expected to see an area of study from an holistic point of view, and not just as a set of limited topics. Doing further reading will also help you to understand complex concepts more completely.

Many of the ideas in this subject guide will be illustrated by examples in *Octave* or *Processing*. It is important for your learning experience that you go through these examples and understand them thoroughly, both for learning about the effect illustrated and also for your own understanding of how to program a computer. You should also use them as starting points to increase your own understanding, by modifying the examples and observing what changes, and maybe also as starting points for developing your own artworks or artefacts, following on from the accompanying pamphlet entitled Portfolio creation.

There is a Sample examination paper for this subject in Appendix A. This covers material from the first volume of the subject guide as well as the material in this volume, and is of a level of difficulty equivalent to the expected standard of the examination for this subject.

Chapter 1

Light and vision

1.1 Introduction

This chapter, and the following one, cover aspects of perception, with the aim of increasing awareness of the role perception has to play both in the creative process and in the viewing of creative artefacts. This chapter covers aspects of visual perception, focusing in particular on colour – including its systematisation to the point that particular colours can be unambiguously specified numerically, essential for computational treatment – and motion, which will have relevance in Chapter 4.

The irony of a chapter on light and vision being printed in black and white is not lost on the author. Furthermore, as will be discussed in the rest of this chapter, neither the perception of colour nor accurate reproduction of colours is a straightforward topic. In particular, there are many ways in which a digital image or a digital display can fail to reproduce faithfully colours perceived or intended; this has the potential to cause confusion.

Additional reading

Dawkins, R. *The Blind Watchmaker*. (London: Penguin, 2006) [ISBN 0141026162 (pbk)].

Feynman, R.P. *Lectures on Physics*. (San Francisco: Pearson/Addison Wesley) [ISBN 0805390456 (hbk set)] Chapters 35–36.

Stokes, M., M. Anderson, S. Chandrasekar and R. Motta. *A Standard Default Color Space for the Internet – sRGB* (1996). Available at `http://www.w3.org/Graphics/Color/sRGB`

1.2 Light and the eye

What we call 'light' is a small portion of the spectrum of *electromagnetic radiation*: waves propagating through electric and magnetic fields. The eye is the primary organ for converting those electromagnetic waves into sensation, responding to waves with wavelengths between 380 nanometres and 700 nanometres (3.8×10^{-7}m and 7×10^{-7}m; *nanometre* is abbreviated as nm. The wavelength of light is also sometimes quoted in ångströms:[1] there are 10ångströms in a nanometre, and visible light has wavelengths between 3800Å and 7000Å. We will use nanometres throughout this subject guide).

Like all wave phenomena, light exhibits certain characteristic behaviours, such as *reflection* (from mirrors, but also from any visible object); *refraction* (when the properties of the medium changes, such as at a water/air boundary or in a desert);

[1] Anders Jonas Ångström (1814–1874), Swedish physicist.

and *scattering* (turning the sky blue). Since the wavelength of visible light is so small compared with most everyday phenomena, the behaviour is less easy to predict from common-sense than other waves; for instance, it is easy to see that water waves can scatter from a rock in the sea, but it is much less intuitive that blue light scatters off air molecules more than red light.

1.2.1 The eye

Because of the remoteness of the scale of light waves from that of easily-observed phenomena, understanding the human perception of light is not straightforward. We begin our path to understanding by examining the way in which light is detected in the human body.

Anatomy of the eye

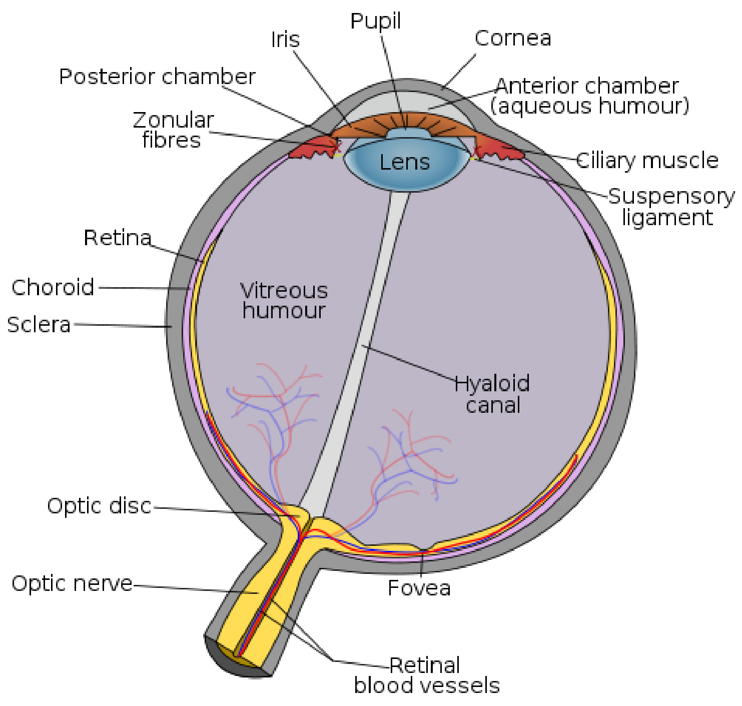


Figure 1.1: A diagram showing the main elements of the human eye. (Source: Wikimedia Commons, released into the Public Domain.)

Figure 1.1 illustrates the main features of the human eye. Light enters through the *cornea*, which refracts and helps to focus it. The *iris* blocks light from travelling further, while the *pupil* allows it through, thus appearing black (as the light that goes through is absorbed by the remainder of the visual system). While the cornea provides most of the optical power, it is fixed, while the *lens* is of adjustable optic power; the *ciliary muscles* can deform the lens, changing its focal characteristics.

The purpose of the focusing is to cause the light to fall onto the *retina*, which contains two principal kinds of photoreceptive cells: *rods* and *cones*. These cells are sensitive to particular kinds of light, firing under some circumstances and not others.

There are two areas of particular interest at the back of the eye: one is the *fovea*, a pit with the highest density of cones on the retina; it is responsible for our sharp central vision, used whenever fine detail is required. The other is the *optic disc*, where the nerve cells joined to the photoreceptors congregate before going into the *optic nerve* towards the brain. It is this area which gives rise to the *blind spot* in human visual perception; there are no useful photoreceptive cells in this area, as all the nerve cells come over the front of the retina, obscuring the light.

There is no reason in principle why the eye has to be this way, with the photoreceptors at the back of the retina and nerve cells coming out to the front; the cells could be designed so that all the nerves came out of the back of the retina instead, which would not cause a blind spot where they joined into an optic nerve. In fact, in cephalopods the photoreceptors are at the front and the nerve cells are at the back of the eye.[2] This issue (and many other aspects of evolutionary biology) is discussed in *The Blind Watchmaker*, which you have already met in the context of generative art (see *Creative computing I*, Vol. 2, Section 6.3.2).

Learning activity

Enter the following code into a *Processing* sketch:

```
// Works for me at a comfortable viewing distance on a 96dpi monitor.
size(1000,480);
smooth(); background(200); noStroke();

fill(0); rect(90,50,20,380); ellipse(800,230,20,20);

fill(20,80,220); rect(65,205,70,70);
```

Then run the sketch; a simple image should pop up. (You may need to adjust the width of the image; it should essentially take up the width of your monitor.)

Next, close or cover your left eye, and align yourself so that your right eye is directly in front of the blue square. Slowly move your head towards and away from the screen, always looking straight at the blue square. With your head at a particular distance from the screen, the black circle will disappear.

Does this work with your other eye? Try rotating the diagram by 180°.

Comments on the activity

This illustrates the effect of the blind spot. You will probably have observed that the blind spot of the right eye is to its right, whereas the blind spot of the left eye is to its left. If both eyes are open, then each eye can see what is in the blind spot of the other, and so the normal field of vision for both eyes is complete.

When the circle disappears, you should have seen that its space was filled with the background (grey) colour. Check that this is the case for different backgrounds. This illustrates that the brain reconstructs the scene from information – inferring that a background is homogeneous unless it receives information otherwise – and that the blind spot can prevent that information from reaching the brain.

---

[2] The octopus is the most well-known member of the cephalopod class.

Photoreceptors: rods and cones

As mentioned above, there are two distinct kinds of photoreceptive cells in the retina: rods and cones. Rod cells are extremely sensitive to light, with the ability to send a signal to the brain if a single *photon* (unit of light) is absorbed. On the other hand, rod cells are slower to fire than cones, with a response time of about 100 milliseconds (abbreviated as ms).

Because of the high sensitivity of rods to light, they are the primary photoreceptors used in dark conditions, such as at night. In addition, the rods are largely absent from the centre of the retina (completely absent from the fovea) and are common on the outside; thus they are used for *peripheral vision*. The rods are sensitive to motion, but have poor spatial discriminating power, which explains why peripheral vision often gives the information that something is moving, but not any detail of what that something is.

The rods have a single kind of pigment within them, which absorbs light in the wavelength range of about 380–640nm. In particular, they are completely insensitive to light of wavelengths above 640nm (a shade of red). This is the cause of the *Purkinje effect*,[3] the name given to the observation that blue objects appear brighter than red ones under darker illumination conditions. Apart from this, rods play little part in colour vision, as they cannot discriminate between different wavelengths of light.

Cone cells, on the other hand, come in three different kinds, each with a distinct pigment; one most sensitive to long-wavelength light, one more towards the medium wavelengths, and one to short-wavelength light, respectively: their peak sensitivies are at wavelengths around 570nm, 540nm and 430nm. The cones are able to respond quickly to a stimulus, and have high concentration in the fovea, giving high-detail central vision. However, they only operate under conditions of bright illumination.

The three different pigments present in cone cells between them permit colour vision: the differently-pigmented cells will fire in different proportions when viewing light of particular frequencies, which allows the brain to attribute a colour to a stimulus. The next section discusses some details of colour vision in more depth.

1.3 Colour vision

1.3.1 Opponent theory of colour perception

The section above describes how the light signal is detected by structures within the eye. However, vision generally and colour vision in particular is not experienced as a sequence of random spots of light or colour flashing on and off in rapid succession; instead, we see whole objects, and whole areas of colour. This tells us that the firing of nerves in response to cone and rod cell excitation is not the end of the story by any means. While the mechanism for detection of light stimulus by the eye is well-understood and uncontroversial, there is no similarly well-understood mechanism for interpreting those detected stimuli as complete visual sensations.

One theory accounting for some empirical observations is known as the *opponent*

[3] Jan Evangelista Purkyně (1787–1869), Czech anatomist and physiologist.

theory of colour vision. The observation that it most clearly explains is the commonly-expressed view that some colours are opposites of each other: red and green are opposites, and so are yellow and blue. By 'opposite' here is meant that there is no such thing as a greenish-red colour or a bluish-yellow; whereas other colours are perceived as mixtures of these opponent primaries.

The proposed mechanism for this opponent theory is that the nerves do not transmit the firing intensities of the different kinds of cone cells directly; instead, the brain receives information about the differences between combinations of these intensities. The physiological effects of opposed colours was investigated in *Theory of Colours* by Goethe;[4] more recent psychological theories use the mechanism of opposition to explain not only colour vision but also other sensations such as emotion and addiction. However, even in the case of colour vision, there is experimental data that suggests that the opponent theory is not precisely true: under certain conditions, it is possible to cause people to perceive a colour which is described as reddish-green.

1.3.2 Colour mixture by addition

In 1853, Grassmann[5] formulated a set of axioms, empirically validated, which are now known as *Grassman's laws of colour perception*. They are (framed in terms of lights which may or may not be mixtures):

- **Additivity**: adding a third light to each of two lights perceived as equal produces equal mixtures. Algebraically: $x = y \Rightarrow x + z = y + z$.
- **Proportionality**: altering the luminances of two equal lights by equal factors produces two equal lights. Algebraically: $x = y \Rightarrow \alpha x = \alpha y$.
- **Transitivity**: equality of light mixtures implies that the equal lights can replace each other in all contexts. Algebraically: $(x = y) \wedge (y = z) \Rightarrow x = z$.

Note that these laws break down at very low luminances, where stimulation of rods is more important than cones; also they are not strictly true over changes in luminance even at ordinary levels, although for practical purposes they are sufficiently accurate over a wide range of illumination and so can be used without significant error.

Grassman's laws imply that if we have two coloured lights representable as mixtures of certain *primaries* (whether those primaries are fundamental to the eye or simply chosen as points of reference), then the mixture of the two colours is also representable as the mixture of those primaries. Specifically, if light X matches $aA + bB + cC$, and light Y matches $a'A + b'B + c'C$, then the mixture $X + Y$ will match $(a + a')A + (b + b')B + (c + c')C$.

This rule of *mixture by addition* – coupled with the assertion that three primaries are enough to match any colour – is the basis for much of digital colour production. Note that in general the coefficients a, b, and c need not be positive; 'subtracting' an amount of colour is equivalent to 'adding' the same amount to the other side. Because of this, it is not possible to produce all colours from wholly **positive** mixtures of colours.

[4] Johann Wolfgang von Goethe (1749–1832), German writer, scientist and diplomat.
[5] Herman Günther Grassman (1809–1877), German mathematician, physicist and linguist.

1.3.3 Colour-based illusions

As well as the uncertainty discussed above as to how colour perception works even in simple cases, there are many interesting effects that can be generated by certain kinds of stimuli. In this section we will look at some illusions, where the perception of colour does not correspond to what is really there.

Pattern-induced flicker colours and Benham's top

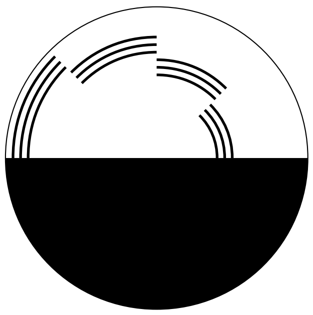
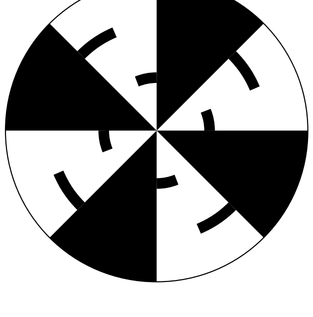


Figure 1.2: A simplified version of the design on Benham's top. When spun, colours are usually perceived, but different colours are seen by different people.

One such is *Benham's top*,[6] which was marketed as a toy in Victorian England. The basic design includes a completely black half circle, while the other half has circular arcs; a simplified version is shown in Figure 1.2. When this disk is spun (at a rate of about 3 to 5 revolutions per second), people perceive colours from the circles described by the arcs, usually pale reds and blues; the colours usually change places if the disk is spun in the opposite direction. These perceived colours are known as *Benham-Fechner colours* (after those who documented them) or *pattern-induced flicker colours* (PIFCs).

Learning activity



The following *Processing* code draws the above circular design, related to Benham's original one, rotating it every frame. Make a sketch and run it, and note what you observe. Try changing the parameters in the sketch, such as the amount of rotation `rot` or the frame rate, and see what changes. What about a design with fewer or more alternating black and white segments?

[6] Charles Edwin Benham (1860–1929), English amateur scientist and polymath.

```
void drawThing(float angle) {
  for(int i = 0; i < 4; i++) {
    fill(0); arc(0, 0, 100, 100, angle, angle+TWO_PI/8);
    fill(255); arc(0, 0, 100, 100, angle+TWO_PI/8, angle+TWO_PI/4);
    noFill();
    arc(0, 0, 70, 70, angle+TWO_PI/8 - 0.1, angle+3*TWO_PI/16);
    arc(0, 0, 40, 40, angle+3*TWO_PI/16, angle+TWO_PI/4+0.1);
    angle += TWO_PI / 4;
  }
}

void setup() {
  size(200,200); stroke(0); strokeWeight(4); strokeCap(SQUARE);
  frameRate(60); smooth();
}

float rot = 0;

void draw() {
  translate(100,100); scale(2); drawThing(rot);
  rot += 0.3;
}
```

Comments on the activity

Because a computer monitor has a certain limiting refresh rate, it is not necessarily possible to have a sufficiently smooth animation for you to see the PIFCs clearly. You may wish to investigate the relative strength of this illusion from the above program running on a digital screen display and a physical spinning disk, stuck for example on a blank CD and spun on a coin. Does the simplified design in Figure 1.2 work better or worse for you on a computer screen?

Cognitive interference and the Stroop effect

The *Stroop effect*[7] is in play when there is interference in the reaction time for performing a task. A particularly striking example is Stroop's *Naming coloured words* task, where the task is to name the colour that a word is printed in.

Learning activity

Use the following *Processing* code to investigate the Stroop effect. How quickly can you name the colours when the words are names of numbers? What about when the words are names of colours? What happens if you use the names of colours in a language that you are less familiar with?

```
String[] numbers = {"ONE", "TWO", "THREE", "FOUR"};;
String[] colours = {"RED", "GREEN", "BLUE", "BLACK"};

class Text {
  String string; color colour;
}

Text[] texts;
boolean useColours = false;

void setup() {
```

[7] John Ridley Stroop (1897–1973), American psychologist and theologian.

```
  size(480,480);
  // You will need the Tools->CreateFont... menu item.
  PFont vera = loadFont("BitstreamVeraSansMono-Roman-48.vlw");
  textFont(vera, 48);
  texts = new Text[21]; makeTexts();
}

void makeTexts() {
  for(int i = 0; i < 21; i++) {
    texts[i] = new Text();
    if(useColours) {
      texts[i].string = colours[int(random(4))];
    } else {
      texts[i].string = numbers[int(random(4))];
    }
    texts[i].colour = (255 << 24) |
      ((random(1) < 0.1) ? 0 : (200 << 8*int(random(3))));
  }
}

void keyPressed() {
  useColours = !useColours; makeTexts();
}

void draw() {
  background(240);
  int i = 0;
  for (int y = 0; y < 7; y++) {
    // compute offsets for justification
    int nchars = 0;
    for (int x = 0; x < 3; x++) {
      nchars += texts[i+x].string.length();
    }
    int space = (440 - 28*nchars) / 2;
    // draw the text
    int xstart = 20;
    for (int x = 0; x < 3; x++) {
      fill(texts[i].colour);
      text(texts[i].string,xstart,60*(1+y));
      xstart += texts[i].string.length()*28 + space;
      i++;
    }
  }
}
```

Spatial correlations and grid illusions

The Hermann grid illusion[8] occurs when a dark background is covered by a light-coloured grid. At the points of intersection between grid lines, dark spots appear transiently, disappearing when they are looked at directly.

A variant on Hermann's grid, called the scintillating grid, was discovered by Elke Lingelbach. The only alteration to Hermann's grid is to draw white circles at the intersections; this causes the perception of a grid of lights flickering (scintillating) on and off. Interestingly, while the Hermann grid illusion can appear with a single intersection, it seems to be necessary to have at least a 3×3 grid for the scintillating effect to be perceived.

[8] Ludimar Hermann (1838–1914), German speech scientist (and, incidentally, coiner of the word *formant*; see Chapter 2).

Learning activity

The code below implements both the Hermann grid and the scintillating grid. Observe both effects, by toggling the boolean variable `CIRCLES`. Does either effect depend on the colours used?

```
size(480,480); background(0); ellipseMode(CENTER);
fill(195,24,0); noStroke(); smooth();
for (int i = 1; i < 8; i++) {
  rect(0, i*60-5, 480, 10);
  rect(i*60-5, 0, 10, 480);
}
boolean CIRCLES = false;
if(CIRCLES) {
  fill(255);
  for (int x = 1; x < 8; x++)
    for (int y = 1; y < 8; y++)
      ellipse(x*60, y*60, 15, 15);
}
```

Comments on the activity

The grid illusions are sometimes explained in terms of a particular neural process of lateral inhibition: in terms of vision, that a stimulus in a particular place tends to decrease the visual response corresponding to nearby locations. This theory is not widely accepted, and there is some evidence against it: the illusions do not tend to appear if the grid lines are curved rather than straight. You may wish to adapt the above code to generate a curved grid and verify this for yourself.

1.3.4 Colour blindness

Many related perceptual deficiencies fall under the general heading of *colour blindness*. The most serious is *monochromacy*, where two or more of the cone pigments are absent from the retina, reducing the space of colours to one dimension; this is exceedingly rare. *Dichromacy*, where one pigment is missing, is more common, affecting about 2% of males (and a much smaller proportion of females): *protanopia* and *deuteranopia* lead to red-green confusion because of a missing long- or medium-wavelength pigment, while the much rarer *tritanopia* leads to yellow-blue confusion.

In addition to these conditions caused by complete absence of pigment, there are also conditions known collectively as *anomalous trichromacies*, resulting from a slight distortion of the pigments in the retina; most important is *deuteranomaly*, which is by far the most common form of colour vision deficiency, affecting approximately 5% of males.

The absence of or anomaly in a retinal pigment leads to some colours being confused; instead of a full three-dimensional colour space, dichromats will not be able to distinguish between any colour on a *confusion line* through that space. This has implications for the design and communication through colour: it is unwise in a design to rely on hue alone, particularly the distinction between red and green hues, because there may be segments of the population who cannot distinguish between the colours: brightness and saturation should also be varied to assist all viewers to see the differences.

Colour blindness is diagnosed using the Ishihara colour test;[9] the test consists of plates with coloured dots arranged over a circle, with arabic numerals in a different colour. People with anomalous colour vision see different numerals from the ones seen by those with ordinary colour vision.

1.4 Colour spaces and profiles

1.4.1 Colour in Processing: RGB and HSB

In working with *Processing*, we have already met, in *Creative computing I*, Vol. 2, Chapter 1, two different colour spaces: the RGB colour space, where each colour is represented in terms of additive composition of three primary colours; and the HSB space, where a colour is identified by values identifying its hue, saturation and brightness. We have also already met the `colorMode()` operator in *Processing*, which alters how colours are specified in *Processing* code.[10]

The RGB and HSB colour spaces are *device-dependent* colour spaces: they do not unambiguously specify a particular colour, as the colour resulting from a particular specification will depend on what equipment is used to display it; other device-dependent colour spaces, not available directly in *Processing*, include subtractive models such as CMY (Cyan-Magenta-Yellow) and HSB variants such as HSL (Hue-Saturation-Lightness). We will introduce *device-independent* colour spaces in Section 1.4.4; these spaces provide the means to specify a particular colour sensation, independently of the device used to display the colour, and so allow the exact reproduction of particular perceptual stimuli. The rest of this section describes in detail the device-dependent colour spaces and the relationships between them.

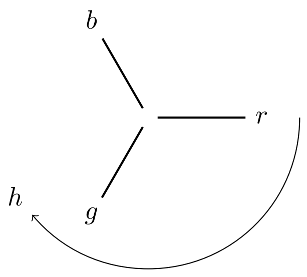


Figure 1.3: Diagrammatic representation of hue in relation to red, green and blue (r, g, b) components of a colour; the hue angle h is the angle from the red axis around a colour circle, and is computed using Equation 1.1.

Let r, g, b be the co-ordinates of a colour in RGB space (with the maximum value normalised to 1 for each); let max be the maximum value of the three co-ordinates

[9] Shinobu Ishihara (1879–1963), Japanese opthalmologist.
[10] Note that the description of `colorMode()` in the *Processing* documentation at the time of writing is misleading; the `colorMode()` operator does not change the interpretation of the colour objects themselves (the signed 32-bit integer) but rather the conversion of a colour specification into such an object of type `color`.

and min the minimum. Then

$$h = \begin{cases} 0 & \max = \min; \\ \frac{\pi}{3} \times \frac{g-b}{\max-\min} \bmod 2\pi & \max = r; \\ \frac{2\pi}{3} + \frac{\pi}{3} \times \frac{b-r}{\max-\min} & \max = g; \\ \frac{4\pi}{3} + \frac{\pi}{3} \times \frac{r-g}{\max-\min} & \max = b; \end{cases} \tag{1.1}$$

gives the *hue angle* from the red, green and blue components (see Figure 1.3; the $\bmod 2\pi$ is there to place the angle in the range between 0 and 2π).

The saturation of a colour in HSB space is essentially how intense the colour itself is, and is computed by:

$$s = \begin{cases} 0 & \max = 0; \\ 1 - \frac{\min}{\max} & \text{otherwise} \end{cases} \tag{1.2}$$

while the brightness is a measure of the overall intensity of the light, and in HSB space is simply given by:

$$\beta = \max. \tag{1.3}$$

To convert back from a hue, saturation, brightness specification to red, green and blue values is the above process in reverse. If h, s and β are the hue, saturation and brightness values, then let i be $\lfloor \frac{3h}{\pi} \rfloor$ (indicating which sixth of the hue circle the hue is in) and f, the fractional part of the hue sextant, be $\frac{3h}{\pi} - i$. Then to compute the (r, g, b) values, compute:

$$\begin{aligned} p &= \beta \times (1 - s) \\ q &= \beta \times (1 - f \times s) \\ t &= \beta \times (1 - (1 - f) \times s) \end{aligned} \tag{1.4}$$

and then assign to (r, g, b) as follows:

$$(r, g, b) = \begin{cases} (\beta, t, p) & i = 0; \\ (q, \beta, p) & i = 1; \\ (p, \beta, t) & i = 2; \\ (p, q, \beta) & i = 3; \\ (t, p, \beta) & i = 4; \\ (\beta, p, q) & i = 5; \end{cases} \tag{1.5}$$

Learning activity

Implement a pair of *Processing* classes, `RGBColor` and `HSBColor`, with fields for red, green, blue and hue, saturation, brightness respectively.

Now implement a pair of functions, `RGB_to_HSB` and `HSB_to_RGB`, which take as argument an instance of the appropriate class and converts it to the representation of the same colour in the other colour space. You will need to define appropriate ranges for each of the member variables.

Comments on the activity

This activity emulates the *Processing* functions `red`, `green`, `blue`, `hue`, `saturation` and `brightness`, which can be used to extract the respective components of colours represented as the `color` datatype (which, as mentioned in *Creative computing I*, Vol. 2, Section 1.2, is simply a signed 32-bit integer or `int`). You should test your implementation by comparing it against the built-in procedures. If you get a discrepancy, but you can't find a bug in your code, try to explain the discrepancy.

Spatial representations of colour spaces

The RGB and HSB colour spaces have natural representations as three-dimensional shapes ('three-dimensional' because there are three *components* needed to specify a colour). The RGB space is most clearly represented as a colour cube: each of the red, green and blue components of the colour space corresponds to one of the axis directions of the three-dimensional space, and so a particular colour's position in this space is specified by a particular amount of red, green and blue. Given this representation, it is possible to express the *colour distance* between two colours, corresponding to the distance between points in this colour space; for colours C and C', if $\Delta r = r - r'$ (and similarly for the green and blue components) then the difference in colours is:

$$\Delta C_{\text{RGB}} = \sqrt{(\Delta r)^2 + (\Delta g)^2 + (\Delta b)^2}. \tag{1.6}$$

For this to be meaningful, the range of r, g, b (e.g. $[0, 1)$ or $[0, 255]$) needs to be specified.

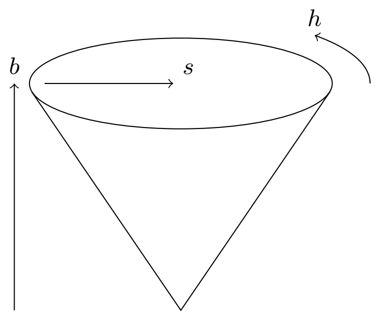


Figure 1.4: The HSB colour space represented as a cone; the directions of hue, saturation and brightness are shown as h, s and b.

The HSB space is most easily represented as a cone: the hue co-ordinate is arranged in a circle; the saturation of a colour increases towards the outside of the cone; and the brightness varies along the cone's axis (see Figure 1.4. This means that a colour position can be represented in co-ordinates as $\{b, bs\cos h, bs\sin h\}$, which means that in this space

$$\Delta C_{\text{HSB}} = \sqrt{(\Delta b)^2 + (bs\cos h - b's'\cos h')^2 + (bs\sin h - b's'\sin h')^2}. \tag{1.7}$$

We will discuss the distances between colours more in Section 1.4.4; for now, be aware that the distances expressed in Equations 1.6 and 1.7 are neither equal to each other nor strongly related to the perception of colour differences. Both of these spaces are useful for computational manipulation, but they do not capture the complexity of relationships between colours.

Learning activity

Using the built-in support for the RGB and HSB colour spaces in *Processing*, construct 3D sketches illustrating the cube and cone representations discussed in this section.

Afterimages and John Sadowski's illusion

After staring at an image for a while, looking at a plain white surface gives the perception of a 'negative' version of the stimulus (where dark regions in the negative correspond to light ones in the original, and *vice versa*). This is known as a *negative afterimage*, and occurs because the cone cells in the retina lose their sensitivity temporarily after being overstimulated. When the attention is turned to a white surface after looking at the image, those cells that previously were firing will be less responsive, while those that were not will respond as normal (and hence will fire more).

Learning activity

Processing has support for inverting an image in the HSB colour space, using the `INVERT` specification to `PImage.filter()`. The following code implements an illusion due to John Sadowski: staring at an inverted image, then at a greyscale version of the original, gives the perception of the full colour image, until the eye moves.

```
PImage orig, gray, invert;

boolean inverted = false;

void setup() {
  orig = loadImage("/home/crhodes/tmp/foo.jpg");
  size(orig.width,orig.height);
  gray = new PImage(orig.width,orig.height);
  gray.copy(orig,0,0,orig.width,orig.height,0,0,orig.width,orig.height);
  gray.filter(GRAY);
  invert = new PImage(orig.width,orig.height);
  invert.copy(orig,0,0,orig.width,orig.height,0,0,orig.width,orig.height);
  invert.filter(INVERT);
  colorMode(HSB);
  invert.loadPixels();
  for (int i = 0; i < invert.pixels.length; i++) {
    float h = hue(invert.pixels[i]);
    float s = saturation(invert.pixels[i]);
    float b = brightness(invert.pixels[i]);
    invert.pixels[i] = color(h, (s+255+255)/3, (b+255)/2);
  }
}

void keyPressed() {
  inverted = !inverted;
}
void draw() {
  image(inverted ? invert : gray,0,0);
  fill(0);
  ellipse(orig.width/2,orig.height/2,2,2);
}
```

The `INVERT` filter inverts the hue in HSB space, whereas the opponent process theory of colour vision would suggest that Sadowski's illusion should be stronger if red is transformed to green and yellow to blue. Investigate which version is more effective for you.

1.4.2 Subtractive colour models

The colour model in *Processing*, and indeed in the vast majority of computer applications, is additive: colours are generated by additively mixing different intensities of primary colours. This is a good model to use when the emission of light can be controlled directly; however, there are many situations where light emission is fixed or not under direct control, and instead the reflection and absorption of light by filters or pigments is used to generate colour.

Conceptually the simplest case of this is the use of filters to let light through selectively; instead of having the starting state being 'no light' and adding to it, filtering usually starts with white light, a mixture of many light wavelengths (for example, from a filament light bulb), and subtracts light components from it (by interposing a coloured film, for instance).

Where in additive mixing the three primaries should correspond to sources spanning as much of the colour space as possible (see Section 1.4.4 below for a more precise statement), the subtractive primaries should correspond to filters **removing** one of the primaries – or, in other words, transmitting a mixture of the other two. Thus a set of three filters which respectively block red, green and blue light act as the primary colours, transmitting respectively cyan, magenta and yellow light.

Then, producing light of one of the additive primaries can be done by applying two appropriate subtractive primary filters in succession: we can produce red light, for example, by applying a magenta filter (transmitting red and blue, but removing green wavelengths from the white light) and then a yellow filter (transmitting red and green but removing blue): the net effect of the two filters is to allow through red light only. The other combinations of two filters can be used to transmit green light and blue light from a white light source.

Subtractive mixing is used in the *process colour* or CMYK model of colour printing. A colour printer has four inks: one ink for each of the subtractive primary colours, and a black ink (known in this context as *key*). To create regions of other colours, the subtractive primary colours are printed on top of each other, effectively combining their subtractive effects; a cyan dot overprinted on a yellow one produces a green dot, just as with the light filters.

A combination of the three subtractive primaries produces black; however, there are a number of reasons why process colour includes a separate black ink. Firstly, inks are not perfect subtractive primaries, and consequently the black produced by their mixture is not necessarily a strong, dark black; it can sometimes be a muddy brown. This is particularly important when printing text, which has the additional requirement of precise reproduction of fine details (such as serifs): to produce those serifs from a mixture of three inks, the *registration* (or alignment) of the three separate coloured images would have to be extremely precise.

As well as these aesthetic aspects to the use of the key ink, there are practical ones: a full mixture of three different inks on the paper can either take too long to dry, or even cause the paper to fall apart. Finally, using a black ink for black, and for darkening colours, will be much cheaper than the mixture of three coloured inks.

While the key ink can be used to darken tones, it obviously cannot lighten them. The way that light colours – and colours not producible by an equal mixture of inks – are printed is through colour mixture by averaging, discussed in the next section.

1.4.3 Other colour mixture models

In the previous sections we have covered mixture by addition, where a compound stimulus is created through the addition at source of two stimuli; observing the mixture produces a sensation distinct in character from the two mixed stimuli. There is a second way in which mixtures can be produced: instead of having a mixed stimulus, the visual system itself produces a mixture by **averaging** different stimuli arriving at the eye.

The first way in which this can happen is through colour mixing by averaging over an area, where regions of different colour are amalgamated into a single colour. The *pointillism* and *divisionism* styles of painting use this feature: many small dots of varied colours are painted, which when viewed from a distance meld into an area of a single colour. Pointillism in particular makes an interesting use of this averaging, by using individual dots of strongly contrasting colours.

This technique, using the eye to mix colours, sees applications in digital displays and in printing. Digital displays with 24-bit colour (8 bits for each of a red, green and blue primary) are considered to be adequate in the colour space resolution, and are now common; displays with lower colour resolution, however, are still in existence, and can be used to reproduce images with a higher colour resolution by *dithering*: to reproduce a colour not exactly representable in the lower-resolution colour space, pixels in the region are coloured so that their average is close to the desired colour.

In printing, a similar technique called *halftoning* is used: the dots printed by a halftoning process can be of different sizes, and are on grids with different orientations. The result is that different amounts of pigments of different colours arrive on the paper, some dots overprinting others wholly or partially and some not; the visual system produces the desired colour by averaging. This phenomenon is also used to make printed colours lighter; assuming that the ink is printed onto white paper, by using less ink (smaller dots) more light of all colours is reflected from that area, which will lead to a lighter average colour.

Learning activity

Use the following *Processing* code as a basis for investigating the colours achievable by dithering.

```
colorMode(RGB,1.0);
for (int x = 0; x < 100; x++) {
  for (int y = 0; y < 100; y++) {
    int parity = (x + y) % 2;
    stroke(parity,1,(1-parity));
    point(x,y);
  }
}
```

Comments on the activity

If you have access to a colour printer, you may wish to investigate how the images that you generate with this sketch are printed, and how you perceive them. Compare the printed output of a dithered colour with the printed output of the closest equivalent colour you can produce on the screen: do the two look the same on paper? Examine the printed output with a magnifying glass.

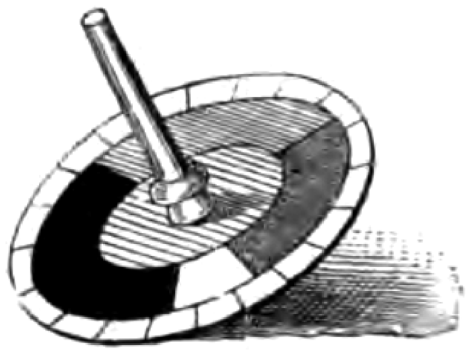


Figure 1.5: An illustration of Maxwell's disc. Three primary colours and the colour being analysed are on the outer ring, and are matched when spun by white and black components on the inner ring, allowing the determination of the colour's composition. (Source: Wikimedia Commons, released into the Public Domain.)

A similar colour mixing phenomenon can occur, but averaging stimuli over time rather than over an area; if given a stimulus of two rapidly-alternating colours, the visual system integrates them, giving the perception of the average colour. This perceptual effect was used in early attempts to systematise colour, using a device called Maxwell's disc[11] (see Figure 1.5).

Learning activity

Investigate colour mixing by averaging on a computer screen. Are there any limitations from using a computer as the medium, compared with Maxwell's disc? What about compensating advantages to the digital medium?

```
boolean yellow;

void setup() {
  yellow = false;
}

void draw() {
  yellow = !yellow;
  if(yellow)
    background(255,255,0);
  else
    background(255,0,0);
}
```

Mixing pigments, for example paints on a palette, is qualitatively different from the mixtures by averaging described above; in general, it is not possible to tell from the colours of two pigments the colour of the mixed pigment. There are two reasons for this: the first is because there are many different absorption or transmission characteristics that will lead to the same perceived colour (this phenomenon is known as *metamerism*), but the colour from a mixture will depend on the details of the spectral characteristics. The second is that the absorption characteristics of the

[11] James Clerk Maxwell (1831–1879), Scottish mathematician and physicist.

mixture depends as well on the physical details of the interactions between individual pigments. Two yellow pigments which appear identical to the eye, can produce distinctly different colours when mixed with a blue pigment: one might produce a bright green, the other a dull grey. The only way of being sure what will be produced is to try the mixture.

1.4.4 Systematisation of colour

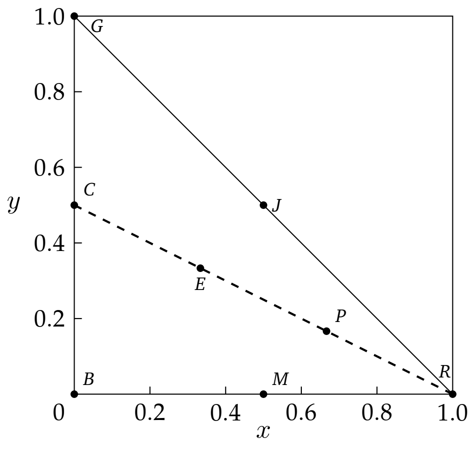


Figure 1.6: A Maxwell triangle representing the colour space of red, green and blue primaries. The secondary colours (cyan, magenta and yellow) are on the edges of the triangle, half way between the primaries (at points *C*, *M* and *J* respectively); white ('equal-energy white') is at point *E*. The dashed line is a *mixture line*, and indicates the positions of the colours in this diagram producable by mixing red and cyan light together; *P* is a pink mixture of red and cyan.

Given three specific additive primary stimuli, we can represent the colour or *chromaticity* of an additive mixture as a point inside a triangle with the primaries at the corners, known as a *Maxwell triangle*; the *gamut* of colours expressible by all possible mixtures of these primaries lies within or on the edge of the triangle (see Figure 1.6). The chromaticity that is represented in Maxwell's triangle is the qualitative aspect of psychophysical colour; the quantity of that colour – corresponding to brightness or *luminance* – is not represented.

To work out the position of a mixture of primaries in the Maxwell triangle, let r, g and b be the amounts of red, green and blue in the mixture. Then the position is at x co-ordinate $\frac{r}{r+g+b}$ and y co-ordinate $\frac{g}{r+g+b}$; for example, if $r = 60$, $g = 80$ and $b = 60$ (a greenish grey), the chromaticity co-ordinates are (0.3,0.4) and the total luminance is 200.

Using this co-ordinate system, we can now specify colours precisely as an additive mixture of specified primaries, produced perhaps in some well-defined physical situation. However, as mentioned in Section 1.3.2 above, it is not possible to find any three stimuli which can produce all visible colours by addition; the gamut of all colours does not form a triangle – instead, to match some colours, it is necessary to **add** primaries to that colour in order to produce something that can be matched with the other primaries.

The *Commission internationale de l'éclairage* or CIE is an international authority on light, colour and illumination; it was established in 1913, and has produced standards for describing colours unambiguously. The colour space in question resembles Figure 1.6, with three primary stimuli, but the primaries themselves do not correspond to visible colours – the gamut of the primaries is larger than that of all colours.

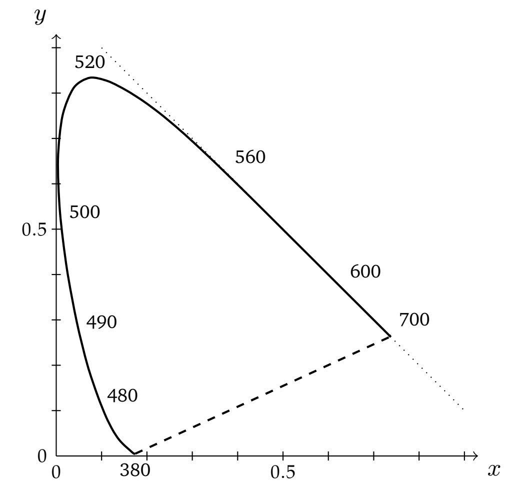


Figure 1.7: The CIE 1931 standard observer on the xy plane. The gamut of visible colours lies within the thick line; all other points in the plane contain impossible negative components of some real colour. The dotted line is the limit of the (imaginary) CIE 1931 gamut; colours from pure wavelengths lie along the thick solid curve (the numbers correspond to the wavelength in nanometres), and the thick dashed curve is the 'purple line': non-spectral colours which are perceptually a mixture of red and blue.

The CIE 1931 colour space selected three imaginary primaries, known as the CIE 1931 *standard observer*. The quantity of the imaginary primaries (or *tristimulus values*) X, Y and Z, then define a specific colour; exactly as with red, green and blue, we can construct a Maxwell triangle for the chromaticity in that colour space, defining $x = \frac{X}{X+Y+Z}$ and $y = \frac{Y}{X+Y+Z}$. The resulting chromaticity diagram for the CIE 1931 colour space is shown in Figure 1.7; any visible colour can be specified by its position in that chromaticity diagram, along with a luminance value. To convert back from chromaticity and luminance (xyY values) to tristimulus (XYZ) values, we use $X = \frac{x}{y}Y$ and $Z = \frac{1-x-y}{y}Y$.

The CIE 1931 colour space is adequate for its intended purpose, which is to specify all possible visible colours unambiguously; as an example of its use, we can now present the colours that people with particular forms of anomalous vision will be unable to distinguish, as in Figure 1.8. However, it does not represent a perceptually uniform space; there is much more resolution in the green area (around 500nm) than the red and blue, even allowing for the greater sensitivity of the eye to greens. This means that the CIE 1931 space does not represent colour differences in a uniform way; it is not possible to use CIE 1931 colour co-ordinates directly to answer questions such as, for example, which of two colours is perceptually closer to a third colour (see Chapter 3 for examples of when one might want to do this).

The CIE LAB (or more properly CIE $L^*a^*b^*$) space is a transformation of the CIE XYZ space into a co-ordinate system where Euclidean distances (see Section 3.3.2) in that

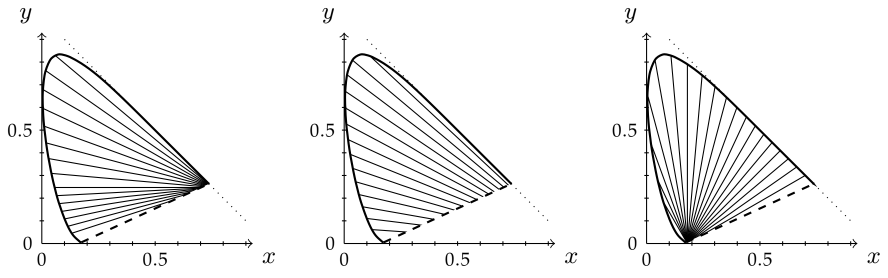


Figure 1.8: Confusion lines in chromaticity space for protanopes (left), deuteranopes and tritanopes (right), illustrating the chromaticity of colours that are indistinguishable for viewers with those forms of anomalous vision. The locations of the points of intersection in chromaticity space are approximately $(0.75, 0.25)$ for protanopes, $(1.0, 0.0)$ for deuteranopes, and $(0.18, 0.0)$ for tritanopes. (Adapted from Judd, D.B. *Fundamental Studies of Color Vision from 1860 to 1960,* Proceedings of the National Academy of Sciences 55:6 (1966).)

system approximately correlate to percieved colour differences. The L^* co-ordinate represents the lightness of the colour, just as the Y co-ordinate does in CIE XYZ, while the a^* and b^* co-ordinates together represent the chromaticity.

To convert from CIE XYZ co-ordinates to CIE LAB,

$$\begin{aligned} L^* &= 116 f\left(\frac{Y}{Y_0}\right) - 16 \\ a^* &= 500\left[f\left(\frac{X}{X_0}\right) - f\left(\frac{Y}{Y_0}\right)\right] \\ b^* &= 200\left[f\left(\frac{Y}{Y_0}\right) - f\left(\frac{Z}{Z_0}\right)\right] \end{aligned} \tag{1.8}$$

where the function $f(t)$ is defined as:

$$f(t) = \begin{cases} \sqrt[3]{t} & t > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{otherwise} \end{cases} \tag{1.9}$$

and X_0, Y_0 and Z_0 are the CIE XYZ tristimulus values of a reference white point, usually the point known as CIE D_{50}, with tristimulus values $X_0 = 96.42$, $Y_0 = 100.00$ and $Z_0 = 82.52$.

Then, given two colours, the colour distance ΔC between them can be computed using:

$$\Delta C = \sqrt{(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2} \tag{1.10}$$

The CIE LAB colour space is used in many computational applications, as it provides both a way of specifying colours precisely and unambiguously, and a way of manipulating colours with predictable perceptual effects. Many file formats for visual data, such as the PDF (Portable Document Format) and TIFF (Tagged Image File Format) support the use of the CIE LAB colour space; image-manipulation programs such as Adobe Photoshop are likewise capable of working within this space.

The transformation to CIE LAB from CIE XYZ is a reversible operation; the inverse transform takes a colour description in $L^*a^*b^*$ co-ordinates and produces XYZ co-ordinates as follows: first define the inverse of equation (1.9) as:

$$f^{-1}(z) = \begin{cases} z^3 & z > \frac{6}{29} \\ \left(z - \frac{4}{29}\right) 3 \left(\frac{6}{29}\right)^2 & \text{otherwise} \end{cases} \tag{1.11}$$

and then:

$$\begin{aligned} X &= X_0 f^{-1}\left(\frac{L^* + 16}{116} + \frac{a^*}{500}\right) \\ Y &= Y_0 f^{-1}\left(\frac{L^* + 16}{116}\right) \\ Z &= Z_0 f^{-1}\left(\frac{L^* + 16}{116} - \frac{b^*}{200}\right) \end{aligned} \tag{1.12}$$

1.4.5 Colour profiles

Now that we have seen a representation of the entire colour gamut perceivable by the eye, it is straightforward to understand the statement that it is not possible to reproduce all colours perceivable by the eye using three primaries of red, green and blue light; the visible gamut does not form a triangle, and so no additive mixture of three real primary colours can possibly span the gamut.

Nevertheless, digital displays **do** use three real primary colours. In order to ensure that these displays reproduce the intended colours from image files, a standard colour profile, sRGB, was developed by Hewlett-Packard and Microsoft.

Converting from CIE XYZ tristimulus values to sRGB involves two transformations. The first is an axis transformation, to convert from the CIE XYZ primaries into specific red, green and blue primaries; we can express this transformation as a matrix multiplication:

$$\begin{pmatrix} R_l \\ G_l \\ B_l \end{pmatrix} = \begin{pmatrix} 3.2410 & -1.5374 & -0.4986 \\ -0.9692 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0570 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{1.13}$$

where the tristimulus values are scaled so that $Y = 1.0$ corresponds to the maximum brightness of the display hardware.

The second transformation adjusts (with *gamma correction*) for the nonlinear nature of brightness on computer monitors and similar digital displays; in Cathode Ray Tube monitors, for example, the physical processes involved in the emission of electrons and the excitation of the phosphors to produce the image on the screen give a perceived brightness that is not linearly related to the input signal, but related by an approximate power-law instead. Liquid Crystal Displays have a very complicated relationship between input voltage and perceived brightness; however, they incorporate hardware to mimic the power-law behaviour of a CRT. If C refers to each of R, G and B in turn:

$$C_{\text{sRGB}} = \begin{cases} 12.92 C_l & C_l < 0.00304 \\ 1.055 C_l^{1/2.4} - 0.055 & \text{otherwise} \end{cases} \tag{1.14}$$

The C_{sRGB} values are clamped to be between 0 and 1, and then scaled to whatever colour resolution is required; for 8-bit colour channels, such as are found in

Processing, Cascading Style Sheet colour specifications, and PNG images (among other uses), the C_{sRGB} values are multiplied by 255.

Apart from the clamping of values to between 0 and 1, the transformation from CIE XYZ to sRGB is reversible; to convert from sRGB co-ordinates to XYZ tristimulus values, first invert the transfer function to yield linear values C_l with:

$$C_l = \begin{cases} \frac{C_{\text{sRGB}}}{12.92} & C_{\text{sRGB}} < 0.04045 \\ \left(\frac{C_{\text{sRGB}}+0.055}{1.055}\right)^{2.4} & \text{otherwise} \end{cases} \tag{1.15}$$

and then invert the matrix multiplication:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{pmatrix} \begin{pmatrix} R_l \\ G_l \\ B_l \end{pmatrix} \tag{1.16}$$

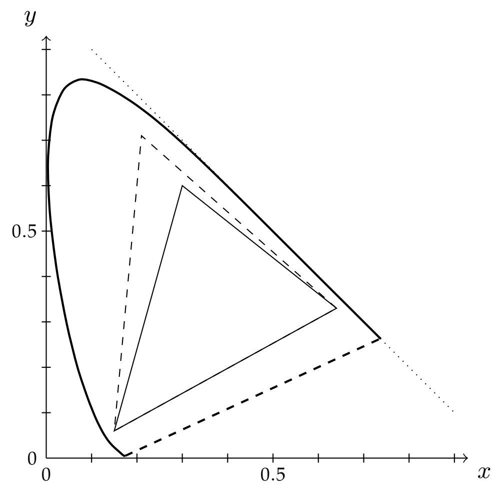


Figure 1.9: A representation of sRGB (solid triangle) and Adobe RGB (dashed triangle) colour space gamuts, relative to the CIE standard observer chromaticity gamut of Figure 1.7.

As well as sRGB, there are other standardised colour spaces, intended for example for professional printing workflows. One such is Adobe RGB, which has a larger gamut of colours than sRGB (compare the dashed triangle in Figure 1.9 to the solid triangle); however, all image manipulation software and printing hardware must be aware of the choice of colour space, otherwise colour manipulation operations will not do what is intended.

1.5 Motion

The perception of motion by the visual system is exploited in the construction of video and animation artefacts, with the illusion of motion generated from the presentation of a succession of static images. This section describes some of the perceptual effects behind the illusion of motion. Regarding the algorithmic detection of motion, note that not everything is fully understood, to the extent that it is difficult to characterise computationally the particular motion that will be perceived from a digital video stream. We will return to this when we discuss multimedia information retrieval in Chapter 3.

1.5.1 Philosophy of motion

Even the concept of motion has certain philosophical difficulties, as it is bound up in the continuity of self over time: to perceive motion, an observer must understand that an object observed twice, once at a later time than the other, is the same underlying object. Zeno's paradoxes[12] illustrate those difficulties, which, even if they have mathematical resolutions, might remain troubling from a philosophical perspective. Zeno is said to have devised his paradoxes in support of the philosophy of his teacher Parmenides, saying amongst other things that change and motion are only illusions.

Two related paradoxes devised by Zeno – the *Dichotomy* paradox, and the paradox of Achilles and the Tortoise – essentially illustrate that an infinite process needs to be achieved before any motion can be deemed to have taken place.

The dichotomy paradox begins by considering motion from a point A to a different point B. In order to perform that motion, a point C halfway between A and B needs to be visited. But to move from A to C, a point D halfway between them must be reached; iterating this argument leads to the conclusion that motion between any two points requires an infinite number of steps.

The paradox of Achilles and the Tortoise is somewhat similar; the scenario involves fleet-footed Achilles racing a tortoise: to make the race fair, the tortoise is given a head start. Now, by the time that Achilles reaches the tortoise's initial position, the tortoise will have moved along a certain distance. For Achilles to overtake the tortoise, he must additionally cover that distance, by which time the tortoise will have moved forward again – and each time Achilles covers the remaining distance, the tortoise inches forward, with the conclusion that Achilles needs to reach the tortoise's current position an infinite number of times before he can overtake.[13]

A third paradox due to Zeno explores a slightly different aspect of the difficulty of motion: the Arrow paradox considers the relationship of motion and time. The scenario involves an arrow in flight towards a target; consider isolating an instant of time in that flight. In that snapshot, the arrow is not in motion – but then how can a succession of instants lead to the arrow progressing in its path? The resolution of this apparent paradox lies in a careful mathematical treatment of infinitesimal quantities; however, the paradox is also related to an illusion allowing the visual perception of motion to arise from a succession of still images, discussed in the next section.

1.5.2 Persistence of vision

Persistence of vision, mentioned in *Creative computing I,* Vol. 1, Section 8.3, can mean one of two things. The strict meaning relates to the fact that the response to a visual stimulus can persist after that stimulus has disappeared; a simple example of this can be seen by moving a luminous object (such as a torch or sparkler) in the dark:

[12] Zeno of Elea (490 BC? – 430 BC?), Greek philosopher.

[13] Achilles and the Tortoise are used as characters in the dialogues in *Gödel, Escher, Bach: an Eternal Golden Braid* by Douglas Hofstadter, a book exploring the relationships between creativity, mind and computation (London: Penguin, 2000) [ISBN 9780140289206].

the light leaves an apparent 'trail' everywhere that the stimulus has been observed in the previous second or so. This occurs because, as discussed in Section 1.2.1, the photoreceptors in the eye have a high response time; the chemical reactions involved in detection of light are not instantaneous, but happen over a period of time.

The second meaning of 'persistence of vision' is the overall processing by the visual system – eye and brain – which allows the perception of motion from a series of still images shown in rapid succession. The persistence of visual response to a momentary stimulus is only one small part of the overall effect of inducing motion where none is present; some other perceptual effects playing their part in the perception of motion – and in particular *beta motion* – are introduced in the next section; we will return to the implications of persistence of vision in the context of animation in Section 4.3.

Persistence of vision has implications for the design of projection apparatus, for example for use in cinemas. There are two rates of importance in such an apparatus. The first is the *frame rate*: the rate at which different images are displayed on the screen; for smooth motion to be perceived, the frame rate should be above about 16 hertz (16 frames per second – modern film runs at 24 frames per second); frame rates lower than this do not give a convincing illusion of smooth motion when rapid movements are being conveyed.

The second rate is the *flicker rate*: the visual system can perceive a regular flicker (between frames) as a distraction even at relatively high frequencies. The flicker rate is at least as high as the frame rate, as there must be one interruption between frames; however, modern projectors are designed to add additional interruptions, to increase the flicker rate (and so decrease its detectability). Typically, the projector shutter is made double-bladed, so that one of the blades obscures the light beam when the projected image is being updated, and the other at the halfway point; for a frame rate of 24Hz, this would give a flicker rate of 48Hz; a triple-bladed apparatus would give a flicker rate of 72Hz.

1.5.3 Motion illusions and Gestalt

In *Creative computing I*, Vol. 1, Section 7.2, several principles of Gestalt psychology were introduced. The Gestalt school of psychology was largely initiated by experiments performed by Wertheimer,[14] researching into the perception of motion. He and his colleagues used very simple stimuli, arrangements of dots switched on and off in particular patterns, to elicit perceptual responses from subjects. Two effects in particular were described by Wertheimer: *beta motion* and the *phi phenomenon*.

Beta motion

Beta motion is believed to be the perceptual illusion responsible for converting a succession of still images (as projected in a cinema, for example) into the perception of motion.

[14] Max Wertheimer (1880–1943), Czech psychologist. His paper *Experimentelle Studien über das Sehen von Bewegung* (1912), Zeitschrift für Psychologie 61, 161–265 is credited with launching the Gestalt revolution.

Learning activity

Type the following *Processing* code into a sketch, and run it. What do you observe?

```
boolean left;

void setup() {
  left = false;
  smooth(); frameRate(3); size(200,200);
}

void draw() {
  background(0); left = !left;
  if(left) {
    ellipse(50,100,30,30);
  } else {
    ellipse(150,100,30,30);
  }
}
```

Alter the sketch so that different stimuli are used instead of the circles: for example, squares, text or an image. Does this affect your perception of the scene? What happens if you use different colours?

If one of the stimuli is made smaller than the other, some people report that the motion is backwards and forwards in addition to side-to-side. Implement this, and see what your perception is.

Comments on the activity

A typical response to this stimulus is along the lines of: 'the circle moves from the left side to the right and back again'.

Phi phenomenon

A separate peceptual illusion of motion was described by Wertheimer, who called it the phi phenomenon or *pure motion*: distinguishable from beta motion by the fact that it was objectless motion: the motion perceived in the phi phenomenon is not that of any of the stimuli, unlike beta motion. It is this observation which was truly novel and serendipitous in 1912, and which kickstarted the Gestalt movement in psychology.

Learning activity

Type the following *Processing* code into a sketch, and run it. What do you observe?

```
int NSTIMULI = 6;
int pos;

void setup() {
  pos = 0;
  smooth(); frameRate(15); size(200,200);
  ellipseMode(CENTER);
}

void draw() {
  background(0);
  int i;
  for (i = 0; i < NSTIMULI; i++) {
    float phase = 2*PI*i / NSTIMULI;
    if (!(i == pos)) {
      ellipse(100+50*cos(phase), 100+50*sin(phase), 30, 30);
    }
  }
  pos = (pos + 1) % NSTIMULI;
}
```

Alter the sketch so that different stimuli are used instead of circles: text, images or different shapes. Does this affect your perception of the scene?

Observe the sketch with different numbers of stimuli: is the effect always clear? How does the number of stimuli affect the *Processing* frame rate necessary to see the effect?

Comments on the activity

Typical response: 'something with the same colour as the background is moving round, obscuring the dots'.

The original phi phenomenon was described in terms of just two dots; however, many people find the phenomenon much clearer with more stimuli.

Many textbooks are unclear over the distinction between beta motion and the phi phenomenon; beta motion is the perceived motion of an object, elicited by successive showing of two related stimuli in different spatial locations, while the phi phenomenon is perceived motion of an unseen object with the visual properties of the scene's background, elicited by implied occlusion of stimuli at a relatively high frequency.

More information about the phi phenomenon can be found at the Magni-phi website.[15]

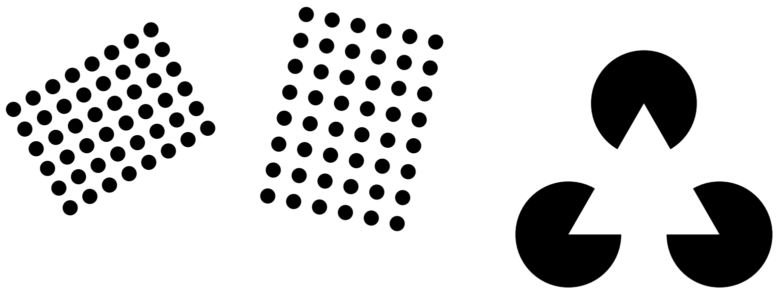


Figure 1.10: Illustration of the Gestalt principles of *proximity* (left) and *closure* (right): in each case, grouping into a structure or inference of a shape occurs without there being a direct stimulus for that inference.

There is no single authoritative list of Gestalt principles of perception; the theory of Gestalt perception has been in existence for about a century, and many distinct effects have acquired the label of 'Gestalt'. However, a conservative list of principles would probably include the effects of *proximity*, *closure*, *similarity*, *continuity* and *common fate*.

[15] `http://www2.psych.purdue.edu/Magniphi/index.html`. See also the paper referenced at that site: Steinman, R.M., Z. Pizlo and F.J. Pizlo (2000) *Phi is not beta, and why Wertheimer's discovery launched the Gestalt revolution: a minireview*, Vision Research, 40, 2257–2264.

The common thread underlying these perceptual principles is the grouping or inference of an entity without having a direct associated stimulus: in Figure 1.10, the grouping into units by proximity and the inference of a triangle's presence over three circles is performed by the perceptual system without any direct evidence. It is exactly this effect – the inference of an object or grouping without any direct stimulus – that is present in the phi phenomenon's objectless motion.

In this chapter we have presented Gestalt perceptual phenomena using visual examples. However, we will find in the next chapter, on the subject of sound, hearing and music, that many of the same principles carry over to the auditory domain.

1.6 Summary and learning outcomes

This chapter introduces elements of visual perception, beginning with a detailed description of how the anatomy of the eye influences the perception of light and colour. The theory of colour perception is covered in some detail, including the opponent theory of colour perception and Grassman's laws of colour perception. The physiological causes of colour blindness, and the prevalence of each of the various forms of colour-related anomalous vision, are discussed in detail.

The notion of a colour space is revised; the device-dependent colour spaces used in *Processing*, along with spatial representations of those spaces are reintroduced and formalized. Subtractive and averaging colour models are discussed, and the notion of colour systematisation is introduced with Maxwell's triangle. The motivation for the CIE imaginary primaries is given, along with the definition of the CIE 1931 colour space and its corresponding chromaticity diagram; the derived CIE LAB space, and the sRGB colour space used on contemporary digital displays, are explicitly related to the CIE 1931 colour space.

The perception of motion by the visual system is discussed, with the beta motion and phi phenomenon visual illusions placing the perception of motion in its historical context with respect to the Gestalt revolution. Persistence of vision is reintroduced in this context, along with a description of the rates at which a cinematic projection device needs to operate to produce the illusion of smooth motion.

With a knowledge of the contents of this chapter and its directed reading and activities, you should be able to:

- describe the anatomy of the eye and how human visual perception is affected by particular anatomical features;
- discuss visual illusions and what they reveal about the visual perception system;
- describe the various kinds of colour blindness and their incidence in the general population;
- understand the spatial representations of the RGB and HSB colour spaces;
- understand the difference between device-dependent and device-independent colour spaces;
- convert colour specifications from one device-dependent colour space to another, and from one device-independent colour space to another;
- understand the uses of particular colour spaces;
- discuss apparent paradoxes and philosophical problems with motion;

- understand the difference between frame rate and flicker rate in the context of cinematography, and how they relate to persistence of vision;
- describe the beta motion and phi phenomenon illusions, and understand their relationship to Gestalt philosophy.

1.7 Exercises

1. Create an interactive, 3D *Processing* sketch to illustrate schematically the structures in the eye: information that you should impart to the viewer should include:
   - two distinct shapes to represent rod and cone cells;
   - three different pigments for the cone cells;
   - the spatial distribution of rods and cones over the retina;
   - the presence of the blind spot.
2. Convert the following XYZ colours into the CIE LAB colour space, and compute the distances between each pair. Which colours are closest together, and which are further apart? Comment on your answer.

XYZ	sRGB
{38.76,42.87,67.20}	(140,180,210)
{40.05,43.39,73.97}	(140,180,220)
{40.86,47.06,67.90}	(140,190,210)
{40.53,43.78,67.29}	(150,180,210)

3. (a) Write a *Processing* function implementing the transformation given in equations (1.15) and (1.16), to convert sRGB values between 0 and 255 into XYZ tristimulus values.
   (b) Write a *Processing* function transforming XYZ tristimulus values to xyY chromaticity co-ordinates.
   (c) Using these two functions, display the colours available to you in *Processing* on a chromaticity diagram. You may wish to include some animated or interactive component to your sketch, to allow the viewer of your sketch to better understand the chromaticity available to them on an sRGB display.
4. Because of the non-linear transfer function in sRGB, a naïve approach to image scaling does not work (see for example `http://www.4p8.com/eric.brasseur/gamma.html` for discussion of this point). Implement a *Processing* sketch which correctly scales down images of even dimension by a factor of 2, by converting each pixel's colour into CIE XYZ space, averaging blocks of four pixels, and converting back into sRGB. Compare the results of your scaling on some test images with the results from commercial or free image manipulation programmes.
5. Design *Processing* sketches illustrating the Gestalt principles of proximity, closure, similarity, continuity and common fate. In each case, illustrate in a manner of your choosing the inferred structure that does not correspond to a direct stimulus.
6. Select a website with a large number of users, and examine the use of colour from the perspective of accessibility: are the colours used sufficiently distinct even for those with anomalous vision? Are there any mitigating factors, such as alternate stylesheets, which could improve the accessibility of such a website?

Chapter 2

Sound, hearing and music

2.1 Introduction

The previous chapter in this volume discussed light and vision, while this chapter includes not just sound and hearing but also some perceptual aspects of **music**. The apparent difference is because there is a qualitative difference between the perception of light and sound: most people agree on the perception of individual light stimuli, while sound is much more sensitive to context.[1] Thus we have to consider combinations of sound stimuli, at which point we are discussing melody or harmony (Sections 2.4 and 2.3.3 respectively).

Although *Processing* has third-party libraries for production and analysis of sound (such as Sonia,[2] introduced in *Creative computing I*, Vol. 2, Chapter 5, and Ess[3]), we will primarily be using *Octave* for the illustrations in this chapter, following *Creative computing II*, Vol. 1, Chapters 4 and 5. It is a worthwhile exercise to translate the *Octave* examples and given here into a *Processing* equivalent, to assist in familiarization of what is possible in that environment.

Additional reading

Handel, S. *Listening*. (Cambridge, Mass: MIT Press, 1989) [ISBN 0262081792 (hbk)].

Howard, D.M. and J. Angus *Acoustics and Psychoacoustics*. (Oxford: Focal Press, 2006) [ISBN 9780240519951 (pbk)].

Levitin, D.J. *This Is Your Brain on Music*. (London: Atlantic Books, 2007) [ISBN 1843547155 (hbk); 0452288525 (pbk)].

Sacks, O. *Musicophilia*. (London: Picador, 2008) [ISBN 0330418378 (hbk); 0330418386 (pbk)].

2.2 Sound and the ear

2.2.1 Sound waves

To understand how sound is perceived, it is necessary to know what sound actually is. Sound propagates through a medium (usually air, but liquids such as water and even solids can support sound waves) as *longitudinal* waves: where the displacement of the medium's elements are in the direction of propagation of the wave (as opposed to *transverse* waves, where the displacements are perpendicular to the wave

[1] Except for those rare individuals with *absolute pitch*.
[2] Available at `http://sonia.pitaru.com/`
[3] `http://www.tree-axis.com/Ess/`

propagation direction). In air, the wave consists of movements of molecules, and so generates alternate compression and rarefaction, or differences in air pressure.

Pressure is measured in pascals[4] (abbreviated 'Pa'), corresponding to a weight of one newton[5] applied to an area of one metre squared. Atmospheric pressure (caused by the weight of the atmosphere on the ground) is approximately 100kPa (10^5Pa), while the amplitude of pressure differences corresponding to a wave that is just audible is about 20μPa (2×10^{-5}Pa).

Learning activity

The following *Processing* code demonstrates the action of a longitudinal wave (such as sound) passing through a medium, represented by weights joined horizontally by springs. The wave is generated by one of the weights being pushed out of position, then let go (at the start of the simulation).

```
class Atom {
  float x; float y; float vx; float vy;
  Atom(int xx, int yy) {
    x = xx; y = yy; vx = 0; vy = 0;
  }
}

float SPRING_CONSTANT;
float INITIAL_DISPLACEMENT;
float FRAME_RATE;
Atom[] atoms;

void setup() {
  INITIAL_DISPLACEMENT = 2; SPRING_CONSTANT = 0.1;
  FRAME_RATE = 5; frameRate(FRAME_RATE);
  atoms = new Atom[10];
  for(int i = 0; i < 10; i++) {
    atoms[i] = new Atom(10*i, 50);
  }
  atoms[1].x += INITIAL_DISPLACEMENT;
}

void draw_atom(Atom a) {
  ellipse(a.x,a.y,2,2);
}

void draw() {
  background(255); fill(0);
  for (int i = 1; i < 9; i++) {
    atoms[i].vx += SPRING_CONSTANT*(atoms[i+1].x - 2*atoms[i].x + atoms[i-1].x);
  }
  for (int i = 0; i < 10; i++) {
    atoms[i].x += atoms[i].vx;
    draw_atom(atoms[i]);
  }
}
```

Enter the code into a *Processing* sketch and run it. What do you observe? What are the visual effects of changing:

- the spring constant;
- the frame rate; or
- the initial displacement?

[4] Blaise Pascal (1623–1662), French mathematician and philosopher.
[5] Isaac Newton (1643–1727), English natural philosopher.

In air, pressure waves propagate at about 340ms^{-1}, independent of their *amplitude* (the maximum displacement from the rest position), *wavelength* (distance between successive compression maxima) or *frequency* (time between repeats). However, the speed of sound waves through other media is different: in water, for example, the propagation speed is about 1500ms^{-1}.

The amplitude of a sound is often measured on a logarithmic scale in decibels; the change in pressure is measured relative to a reference level (frequently a notional threshold of human hearing), taking the base-10 logarithm of the ratio of the intensities, and multiplying the result by 20 to give the *sound pressure level* (SPL):

$$\mathrm{SPL} = 20\log_{10}\frac{p}{p_{\mathrm{ref}}} \tag{2.1}$$

This logarithmic scale is both convenient for expressing the range of amplitudes in sound waves (most sound lies within the range 0–100dB), and approximates how people hear loudness (see Section 2.2.4 below).

Since sound is a wave phenomenon, it will undergo various effects in common with other waves (such as light waves, see Section 1.2):

- *refraction* when the medium characteristics change;
- *diffraction* or scattering by small objects;
- *reflection* from hard surfaces; and
- *interference* between sound waves.

These effects have implications in the design of musical instruments and in the acoustic properties of rooms and buildings, but are beyond the scope of this subject guide: details can be found in the additional reading materials suggested at the start of this chapter.

2.2.2 The ear

Figure 2.1 illustrates the basic anatomy of the human hearing system, which can be loosely divided into three sections: the *outer ear*; the *middle ear*; and the *inner ear*.

Outer ear

The outer ear consists of a flap of skin, called the *pinna*, the *auditory canal*, a 3cm-long tube, leading to the *tympanic membrane* or eardrum. The primary function of the pinna, together with the opening into the auditory canal, is to assist the hearing system to deduce the direction of a sound source.

The auditory canal itself acts as a resonant cavity, amplifying in particular those frequencies close to its *resonant frequency* of 4kHz. The incoming pressure waves are then propagated towards the tympanic membrane, a light and elastic structure whose function is to convert the sound pressure oscillations into mechanical vibrations which can be processed by the rest of the hearing system.

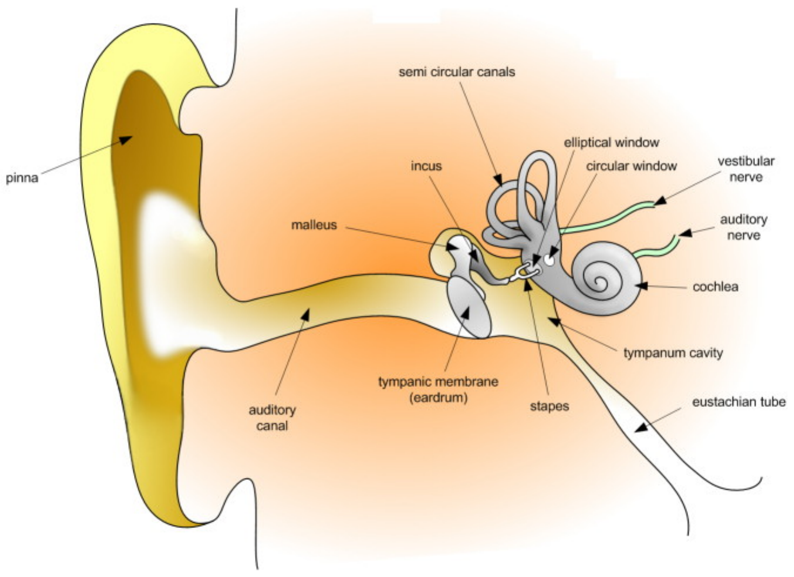


Figure 2.1: A diagram showing the main elements of the human ear. (Source: Wikimedia Commons, released into the Public Domain)

Middle ear

The middle ear is made up of the *ossicles* (small bones): three small structures, known as the *malleus* or hammer, *incus* or anvil and *stapes* or stirrup. Their function is to transmit and amplify the mechanical vibrations of the tympanic membrane to the inner ear, to which they are joined at the oval or *elliptical window*.

The hammer is fixed to the middle of the eardrum, and usually also acts as a single unit with the anvil, rotating to move the stirrup as if by a piston, thus transmitting the acoustic vibrations to the elliptical window. This is achieved with minimal loss of energy by two means: firstly, the hammer and anvil are of different lengths, so there is a lever effect in their action; secondly, the area of the tympanic membrane is larger than the elliptical window, and so energy is effectively focused by the ossicles into the cochlea.

In order to prevent against damage to the hearing system by loud sounds, the middle ear contains muscles which decrease this efficient transfer of energy given a stimulus of high intensity, providing energy attenuation to protect the sensitive systems of the inner ear.

Inner ear

The inner ear is also known as the *cochlea*, a structure shaped like a shell, whose function is to convert the mechanical oscillations transmitted to it into nerve firings which can be interpreted by the brain. The cochlea is filled with liquid (*perilymph fluid*) which is essentially incompressible; the action of the stapes on the elliptical window causes the fluid to move round the cochlea all the way to the circular window, which can move to compensate for the elliptical window's movement.

Within the cochlea is an inner channel, the *scala media*, whose walls are made up of membranes: *Reissner's membrane* and the *basilar membrane*. These are displaced by travelling waves set up by the motion of the stapes; the basilar membrane is responsible for the frequency analysis of sounds: the membrane is tapered, which means that different regions on the membrane respond best to different frequencies; the different regions are attached to hair cells and associated nerves, which fire on displacement; the nerves form a bundle known as the *auditory nerve*, which leaves the cochlea for the brain.

The *eustachian tube* is not directly related to the hearing system; it is connected to the sinuses, and is responsible for regulating the pressure within the ear. It is this channel which causes ears to 'pop' given a sufficient change in pressure, for example on aeroplanes or in tunnels, or by blowing through the nose while pinching it shut.

2.2.3 Sound source direction

How do we perceive the direction from which a sound is coming? The signal arriving at both ears is used, and from the differences between those signals (which are a result of the spatial separation – so that the path taken by the signal from the source to each of the ears is different) the brain can resolve the direction of the source.

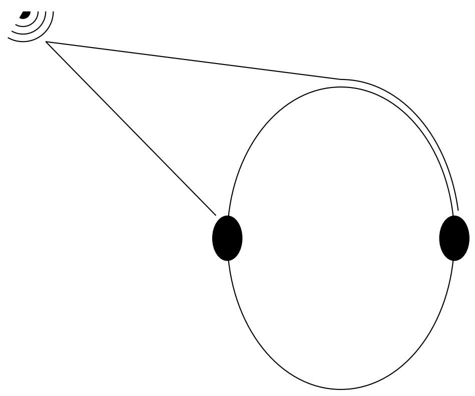


Figure 2.2: Illustration of the path difference between sound waves reaching the ears; the difference in path length causes an offset in the *phase* of the wave at each ear, dependent on the angle and distance of the sound source and the wavelength of the signal; in addition, diffraction by the head causes weakening or *attenuation* of the wave. These cues are processed by the auditory system to help determine the location of a sound source.

There are two basic cues used by the brain for this: the time difference between the arrival of signals, and the intensity difference. Considering the time difference first, think of a sound source on the left side of the head (see Figure 2.2): the sound wave can travel directly to the left ear, but has to travel around the head to the right ear. This time difference is of the order of 0.5ms – a small delay, but enough to resolve direction of sounds at low frequencies.

High-frequency sources cannot have their direction resolved by time difference, because once the path difference causes a phase shift of more than 180° in the signal, ambiguities arise. The second cue for direction comes from the fact that the intensities of the signal at the ears will be different: the signal following the path travelling around the head will be attenuated, and the amplitude difference between the signals is used by the brain to deduce the direction. This cue works best at high frequencies, because for long-wavelength (low frequency) signals the head is not a significant obstacle.

Additional detail is provided by ridges on the ear's *pinnae* – which cause interference patterns in sounds – and by small movements of the head, which cause changes in the relative time and intensity differences of the two ears' signals. This last detail is defeated by sound sources which move with the head (such as headphones); given stimuli from such sources, the resulting perception is of a sound which comes from **inside** the head – since there are no relative changes in intensity or phase with head movement.

Learning activity

Run the following code in *Octave* and listen to the sound produced on a pair of headphones. What do you perceive? What if you actively move your head while listening?

```
octave:> l = [0:1/8000:1] .* 1/2 * sin(2*pi*440*[0:1/8000:1]);
octave:> r = [1:-1/8000:0] .* 1/2 * sin(2*pi*440*[0:1/8000:1]);
octave:> sound([l;r])
```

Comments on the activity

The sensation from listening to this sound is something like having a sound travel through the head from right to left. This effect can be used to simulate stereophonic listening on headphones, in a technique called *binaural stereo*; other strategies include replaying the sound recorded on microphones placed far apart (*delay stereo*) or pointing in different directions (*intensity stereo*), using the cues for direction discussed above.

2.2.4 Loudness

In Section 2.2.1, we introduced the measurement of the intensity of a sound wave through the sound pressure level (Equation 2.1). The perception of *loudness* of a sound, while related to the amplitude of the incident wave, does not have a simple, direct relationship, though there are rules of thumb which are good enough for most purposes.

The first approximation to loudness is essentially that an increase in the SPL value of 10dB corresponds roughly to a doubling in loudness. Table 2.1 shows a series of sounds with their corresponding pressure levels. However, in fact the perceived difference in loudness depends both on the frequency of the sounds and on their amplitude. The effect of the outer ear's resonance in the acoustic cavity, described in Section 2.2.2 is to increase the sensitivity of hearing at frequencies around 3–4kHz, and the hearing system decreases sensitivity at some other frequencies; the amount of decrease is different at different amplitudes, which means that sounds can change in relative loudness at different intensity levels.

Sound	**Intensity / dB**	**Notes**
Threshold of hearing	0	
Breeze through leaves	10	Just audible
Empty hall	20	
Office noise	50	
Normal conversation	60	
Heavy traffic at 1m–10m	85	Damaging (long-term)
Nightclub dance floor	110	
Jet take-off at 100m	120	Damaging (short-term)
Threshold of pain	130	
Atmospheric limit	190	

Table 2.1: Common sounds and their approximate SPL values in decibels, with reference to the threshold of hearing, a pressure wave of amplitude 20μPa being at 0dB.

2.3 Frequency, pitch and harmony

This section explores the relationship between the frequency of a sound wave to the musical concepts of pitch (for a single note) and harmony (the musical effect caused by several notes sounding simultaneously).

Learning activity

In this activity, you will investigate the superposition of sinusoidal signals whose frequencies have particular relationships. Firstly, investigate the result of superposing two sinusoids with relatively close frequencies:

Using the following *Octave* code, generate two sinusoids with frequences 20Hz and 17Hz (assuming a sample rate of 400Hz):

```
octave:> t = [0:1/400:1];
octave:> x = sin(2*pi*20*t);
octave:> y = sin(2*pi*17*t);
```

Next, visualise the superposition, for example with the following octave commands:

```
octave:> plot(t,x);
octave:> plot(t,y);
octave:> plot(t,x+y);
```

Note that the reason for choosing such low frequencies and such a low sample rate is to make it easy to visualise the effects in this way.

Now change the frequency of the y signal to 11Hz: what differences and what similarities do you see? What about when y's frequency is altered instead to 10Hz?

Finally, listen to analogous signals. You will need to increase the frequencies and sample rate for effects to be audible: for example, take x to be a signal at 440Hz, y to be at 436Hz (then 225Hz, then 220Hz), and the sample rate to be at least 8000Hz. One expression for the sum of two sinusoids is:

$$\sin(A) + \sin(B) = 2\sin\left(\frac{A+B}{2}\right)\cos\left(\frac{A-B}{2}\right) \qquad (2.2)$$

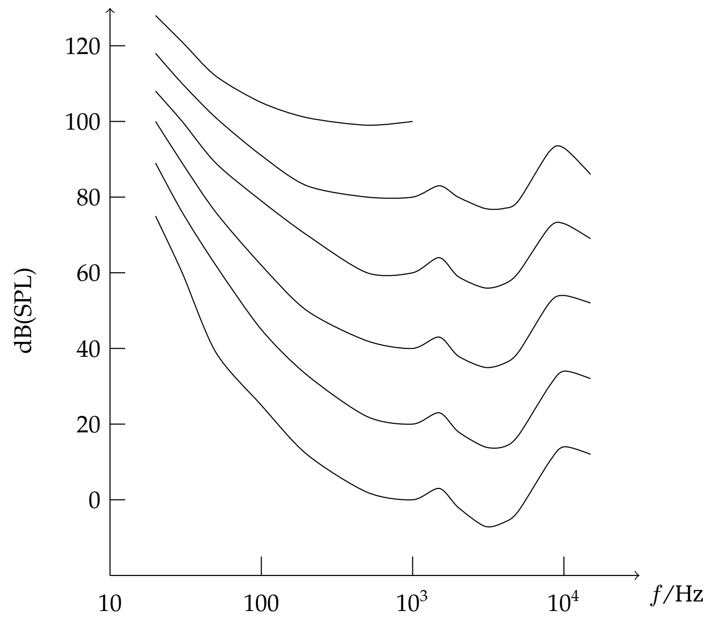


Figure 2.3: Equal loudness curves (data after ISO 226:2003 *Acoustics – Normal equal-loudness-level contours* standard) across different frequencies. Note the clear drop in Sound Pressure Level needed to achieve a particular perceived loudness at frequencies around 3–4kHz.

Can you explain any of your observations in the light of this expression?

Comments on the activity

Some sensations that you will perceive from these sounds will be pleasant, and some unpleasant. The implications of this for musical harmony are discussed further in Section 2.3.3 below.

When two sinusoids with frequencies that are close to each other are superposed, the phenomenon known as 'beating' occurs: the result is a vibration at the average frequency, modulated by a vibration at the frequency **difference**. This is one cause (but not the only one) of dissonance.

2.3.1 Fourier series and the spectral domain

The signals created in the activity above are in the *time domain*, where the evolution of the pressure difference at a point is represented as a time-varying signal. It is also possible to represent signals in the *frequency domain*, drawing from the fact (introduced in *Creative computing II*, Vol. 1, Chapters 3 and 4) that any signal at all can be represented as the sum of sinusoidal signals with the right amplitudes and relative phases. This decomposition of signals into sums of sinusoidal is called Fourier analysis.[6]

For example, we can construct a square wave by adding together odd harmonics

[6] Jean Baptiste Joseph Fourier (1768–1830), French mathematician.

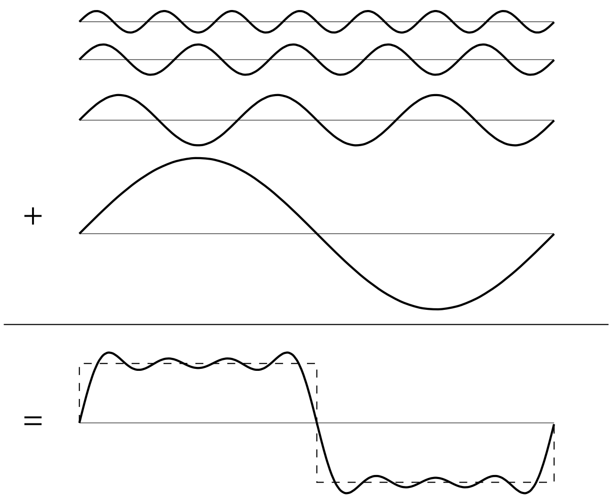


Figure 2.4: Illustration of the construction of a square wave signal from sinusoidal Fourier components.

$\sin(2\pi(2n-1)t)$, each with amplitude $\frac{1}{2n-1}$, illustrated graphically in Figure 2.4 with the first 4 elements; the more components we add, the closer the resulting signal is to an exact square wave. While the time-domain representation of a square wave is a function $f(t)$ which takes alternately values of 1 and -1, the frequency-domain representation is a function $\tilde{f}(\omega)$ which is zero everywhere except at the integer multiples of the fundamental 1, 3, 5 (and so on) where it takes values 1, $\frac{1}{3}$, $\frac{1}{5}$.

2.3.2 The harmonic series, notes and scales

A real musical note perceived to have an unambiguous pitch corresponds to a pressure wave which repeats regularly, at some *fundamental frequency*; if the period in time before the sound wave repeats is τ the fundamental frequency f_0 is given by

$$f_0 = \frac{1}{\tau}; \tag{2.3}$$

when τ is measured in seconds, the fundamental frequency is measured in hertz (abbreviated as 'Hz'): one event per second has a frequency of one hertz.[7]

In addition to the wave vibration at the fundamental frequency, there will be other vibrations present; real musical instruments do not produce a single pure sinusoidal signal. These other vibrations are at integer multiples of the fundamental frequency, and the different patterns of intensity in these *harmonics* give rise to the feeling of different acoustic texture or *timbre* – and assist in the perceptual identification of single instruments.

We can represent the wave vibration produced by a musical instrument playing the note with fundamental frequency f_0 by the following sum:

$$s(t) = \sum_{k=1}^{\infty} [a_k \cos(2\pi k f_0 t) + b_k \sin(2\pi k f_0 t)] \tag{2.4}$$

[7] Heinrich Rudolf Hertz (1857–1894), German physicist.

in exactly the same way as a periodic signal was represented in a Fourier series. The oscillations at frequencies above the fundamental are sometimes referred to as *overtones*: the second harmonic, with frequency $2f_0$, is the first overtone – the first tone over the fundamental.

f_0 **ratio**	**Interval name**
2:1	octave
3:2	perfect fifth
4:3	perfect fourth
5:4	major third
6:5	minor third
5:3	major sixth
8:5	minor sixth

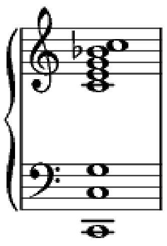


Table 2.2: Musical intervals from the harmonic series: the ratio between the fundamental frequencies of two adjacent harmonics in the series and the corresponding interval is presented in the top portion of the table. The bottom portion shows intervals formed from the others by *inversion*, by doubling the lower fundamental frequency (equivalent to moving the lower note up an octave).

The integer harmonics of a fundamental are of great importance in Western music, forming the basis of the musical system in use for over a thousand years; notes whose fundamental frequencies are related by a small integer ratio are harmonically related, and their use either simultaneously or in sequence is extremely common. Table 2.2 summarises frequency ratios commonly found in music, and the name given to that interval.

The *octave* in Western music is a special interval, because notes separated by an octave are perceived as being the same note. This phenomenon of octave equivalence gives rise to the possibility of forming *scales* of notes within one octave.

Pythagorean harmony As was mentioned in *Creative computing I*, Vol. 1, page 7, a scale based entirely on the ratios 3:2 (the interval of a perfect fifth) and 2:1 (the octave) was devised by Pythagoras and is known as the Pythagorean scale. This scale is derived essentially from the fact that, after ascending in intervals of a fifth (frequency ratio of 3:2) twelve times, the note arrived at is very close to the note that would have been reached by ascending seven octaves. Mathematically:

$$\left(\frac{3}{2}\right)^{12} = 129.75 \approx 128 = 2^7. \tag{2.5}$$

To form the Pythagorean scale, first choose a starting note; then, ascend by a fifth from the previous note in succession, and descend by an octave whenever the note ascended to is more than one octave away from the starting note. This process is illustrated in Figure 2.5.

Equal temperament The Pythagorean scale, while successful for a long time, presents some practical problems, which became more acute as the range of intervals used in music expanded. Eleven of the frequency ratios between notes one fifth apart in the Pythagorean scale are the exact 3:2 ratio which leads to a pleasant sensation when the two notes are played together. However, the twelfth is 2.96:2,

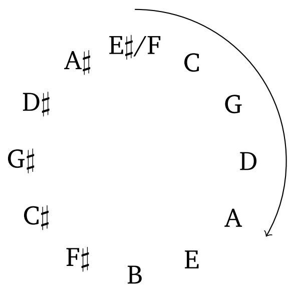


Figure 2.5: The circle of fifths. The Pythagorean scale is formed by identifying the last note, E♯, with the first, F, ignoring the discrepancy of about 1% in their fundamental frequencies. This discrepancy is known as the *Pythagorean comma*.

which is noticeably different, and produces unpleasant rough sensations in the listener. Additionally, while intervals of a perfect fifth are exactly represented in the Pythagorean scale, other intervals (such as the thirds and sixths; refer to Table 2.2) are quite distant from their exact equivalent.

The latter problem was partially resolved by *just intonation*, beyond the scope of this subject; the former was addressed by altering the scale to the *equal tempered* scale: rather than by ascending 11 pure fifths and one fifth adjusted by a comma, the equal tempered scale is formed by ascending 12 exactly equal intervals, each interval being a fifth adjusted by one twelfth of a comma.

The frequency ratio of that adjusted interval is straightforward to calculate, since we must have that 12 of the adjusted interval ϕ be exactly equivalent to 7 octaves:

$$\phi^{12} = 2^7 \Rightarrow \phi = \sqrt[12]{2^7} \approx 1.498. \tag{2.6}$$

This frequency ratio is sufficiently close to 3:2 that this does not cause unpleasant beating sensations, but is perceived as being close to a pure interval. Because all the frequency ratios between fifths are the same, the scale formed from this basis also has a constant frequency ratio between adjacent notes: $\sqrt[12]{2} \approx 1.059$, the equal-tempered *semitone*.

This equal-tempered tuning system has been in use since the 19th century, and is very common in Western music today. During the Renaissance and Baroque periods, many different methods of tuning the scale, or *temperaments*, were in use, including *well temperament*, used by J.S. Bach[8] for his ‘Well-Tempered Clavier’.

2.3.3 Harmony

Music from different cultures varies considerably along many dimensions; different kinds of music are built from different appreciations of pitch, rhythm, instruments and units of melody and harmony. Additionally, musical appreciation is not static, but has changed over history, with what was previously considered avant-garde becoming part of the mainstream. It is, however, possible to suggest reasons for particular ways of appreciating musical elements; in this section, we examine chords of the Western musical tradition from the perceptual point of view.

[8] Johann Sebastian Bach (1685–1750), German composer and organist.

As a simplification, the chords under consideration will be restricted to two notes, each with their own set of harmonics of the harmonic series. From that point of view, and viewed in only the coarsest terms, the history of Western harmony can be summarised by saying that as time went on, it became acceptable to use intervals in a harmonic ratio approaching unity closer and closer. Initially, only notes in unisons and octave ratios were permitted; to a Western ear, unisons and octaves sound like the same underlying note. Organum, the first Western harmonic system, initially used exclusively pairs of notes whose fundamental frequencies were in 3:2 and 4:3 ratios. At the transition between the mediaeval period and the Renaissance, around the year 1400, chords with frequency ratios 5:4 and 6:5 begin to be used as *consonances* (pleasing intervals) rather than *dissonances* (unpleasant or harsh sounds). As the centuries passed, chords with a frequency ratio progressively closer to 1:1 become more and more common, with contemporary Western music using even microtonal intervals (corresponding to the use of harmonics above the 16th).

Consonance and dissonance

As discussed in Section 2.2.2, the basilar membrane is made up of regions with particular characteristic resonant frequency ranges, attached to hair cells which (on vibration of the membrane region) cause electrical signals to be transmitted through the auditory nerve to the brain.

A particular effect of this architecture is that unpleasant sensations will occur if two sinusoidal components with close (but not the same) frequencies are simultaneously heard. The degree of dissonance from a chord made up of a single interval can be assessed by comparing the frequencies of the various harmonics present. The more harmonics that are close together but do not coincide precisely, the more dissonant the chord.

Furthermore, the closeness of harmonics necessary to cause a perception of dissonance is relatively less at lower pitches; the bandwidth of each region on the basilar membrane does not increase linearly with frequency, but rather:

$$\text{bandwidth/Hz} \approx 6.23 \times 10^{-6} f^2 + 9.339 \times 10^{-2} f + 28.52 \tag{2.7}$$

for the central frequency f expressed in hertz. Therefore, the quality factor (Q, from *Creative computing II*, Vol. 1, Section 5.1.3) of regions on the basilar membrane increases with increasing frequency, rather than being constant); this means that the same musical interval will sound more dissonant if the notes making it up are lower in pitch.

Real instruments, as discussed in Section 2.3.2 above, have different profiles of harmonics; therefore, the timbre of the performing instruments will affect the judgment of how dissonant or consonant a particular interval is, because the more energy there is in spectral components making up dissonances, the more the dissonance is noticeable.

For example, Table 2.3 shows the frequencies from the harmonic series present in intervals of a perfect fifth and a major third above a lower note of 240Hz. The interval of a perfect fifth causes minor dissonant sensations from the interaction of the fourth harmonic of the lower note with the third harmonic of the upper note (960Hz and 1080Hz in this case), but all other intervals are either consonant or unisons. By contrast, the major third has strong dissonances between the fourth harmonic of the lower note and the third of the upper (960Hz and 900Hz), and between the sixth harmonic of the lower and fourth of the upper (1440Hz and 1500Hz); therefore, the major third is perceived as more dissonant than the perfect fifth.

Lower note / Hz	**Perfect fifth (3:2)** / Hz	**Major third (5:4)** / Hz
240		300
480	360	
720	720	600
960	1080	900
1200		1200
1440	1440	1500

Table 2.3: Harmonic series frequencies present in chords of a perfect fifth and a major third above a note at 240Hz. Frequencies from the upper note are placed in the row corresponding to the nearest frequency from the lower note.

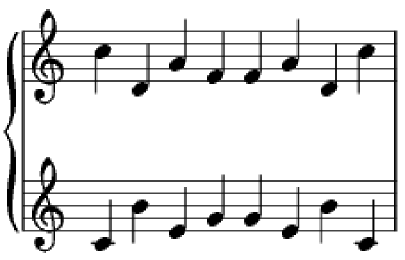
In addition to these effects of consonance and dissonance, a perceptual effect known as *frequency masking* (or *simultaneous masking*) can cause one sound to obscure another; a quieter tone with similar frequency to the masking stimulus can be completely obscured, because they excite similar regions on the basilar membrane in the inner ear. As a rule of thumb, signals which are close in frequency but have about one-tenth of the amplitude (20dB quieter) are masked, and are not perceived. This has implications for compression of audio, discussed in Section 2.6.2 below.

2.4 Melody

A melody is a (usually continuous) time sequence of tones which is perceived as a single musical entity. Successful melodies are highly memorable, and the question of what makes a memorable melody (and even how musical memory functions) is a topic of active research. This section covers some basic aspects of melodic perception: the division of multiple tones into streams, and the cues afforded by timbre.

The following activity illustrates the processing of auditory stimuli into high-level musical streams, even going against low-level cues for inferring the position of the sources of the stimuli.

Learning activity

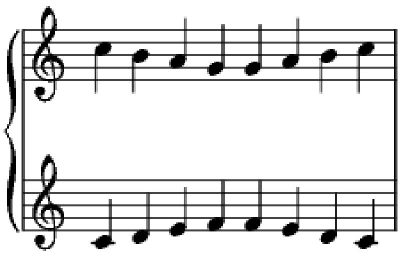


The *Octave* code below generates a stimulus corresponding to the music above, constructed by Diana Deutsch for an experiment in the 1970s. The stimulus consists of an ascending and a descending C major scale, but with notes from each alternated in left and right ears.

```
octave:> fs = 261.62*(2.^(1/12)).^[0:12];
octave:> t = [0:1/8000:0.5-1/8000];
octave:> lt = \
> sin(reshape(2*pi*t'*fs([13,3,10,6,6,10,3,13]),1,32000));
octave:> rt = \
> sin(reshape(2*pi*t'*fs([1,12,5,8,8,5,12,1]),1,32000));
octave:> sound([lt;rt]);
```

Comments on the activity

The most common response to this stimulus is shown below, with the high notes in the left ear and the low notes in the right.



In some cases, the locations of the two responses are reversed, but in the majority of cases the stimulus is separated into two streams as above.

By way of explanation, we can relate this effect to Gestalt principles of perception (see *Creative computing I*, Vol. 1, Chapter 7 and Chapter 1 of this subject guide); in particular, applicable principles are those of *continuity*, *proximity* and *similarity*. In this instance, it seems that the principles of proximity and similarity, grouping 'high' and 'low' tones separately, override the principle of continuity, which would suggest the perception of two separate scales, one ascending and one descending.

The matrix manipulation to make this example work also deserves some explanation. `t` is a row vector of 4,000 elements, so `t'` is a column vector of 4,000 elements, each successive one $\frac{1}{8000}$ greater than the previous. `fs` is a row vector with the frequencies of the equal-tempered scale; `fs([1,12,...])` creates a row vector with the first and twelfth elements of `fs`, so `t'*fs([1,12,...])` corresponds to the matrix multiplication:

$$\begin{pmatrix} 0 \\ \frac{1}{8000} \\ \frac{1}{4000} \\ \vdots \end{pmatrix} \cdot \begin{pmatrix} 261.62 & 493.87 & \dots \end{pmatrix}$$

which results in a 4,000×8-element matrix in the above code. This is converted into a 32,000-element vector with the values following corresponding to each frequency following each other using the `reshape` operator, before passing this vector to the `sin` operator to generate the sinusoidal tones you hear.

The following activity illustrates how timbre can be used to give cues to the perception of melodic entities.

Learning activity

Through the use of timbre, it is possible to fool the ear into hearing a continuously-ascending series of notes. The following *Octave* session will produce such an illusion, constructed in a similar manner to the tones used by Roger Shepard in auditory experiments (known as *Shepard tones*).[9]

```
octave:> global t = [0:1/8000:0.5-1/8000];
octave:> global fs = 261.62*(2.^(1/12)).^[0:12];
octave:> function amplitude = amplitude(f)
> amplitude = max(1/4 - 1/25*(1.5-log2(f/130.8)).^2,0);
> end
octave:> function st = shepard(n)
> global fs;
> global t;
> st = amplitude([1/4,1/2,1,2,4,8]*fs(n)) * \
>                sin(2*pi*fs(n)*[1/4,1/2,1,2,4,8]'*t);
> end
octave:> scale = zeros(1,4000*12);
octave:> for i = (0:11)
> scale(1,i*4000+1:(i+1)*4000) = 0.4*shepard(i+1);
> endfor
octave:> sound([scale,scale,scale,scale])
```

Listen to the sounds produced by by this session. Do you hear a continuously-ascending scale? If not, where does a discontinuity occur? Does the illusion work better, worse or the same if the scale is descending instead of ascending?

Comments on the activity

This continuously-ascending scale has an obvious direct parallel with illusions produced by M.C. Escher (1898–1972) and Oscar Reutersvärd (1915–2002). Escher's lithograph 'Ascending and Descending' (1960) is very well-known; less well-known is the impossible staircase design by Oscar Reutersvärd, predating Escher's design by at least 10 years.

You may be interested in attempting to convert the scale in the activity above to a continuous sweep, rather than being divided up into 12 discrete steps.

To help understand the above code, `t` and `fs` respectively are the sample times for an individual scale step, and predominant frequencies for each step; 261.62Hz is the frequency of middle C. The `amplitude` function scales the presence of different harmonics (at $\frac{1}{4}$, $\frac{1}{2}$, 1, 2, 4 and 8 times the predominant frequency), while the `shepard` function generates the sound corresponding to an individual step. The individual sounds are then collected into the array `scale`, played multiple times by the *Octave* `sound` function.

2.5 Rhythm

Rhythm is a complex, human, phenomenon; it encompasses many different levels, and involves the construction of intricate structures from simple sequences of stimuli; a multifaceted rhythm emerges from the interactions between different levels of perception. As with melody, rhythms can be highly memorable, and little is known about the mechanisms or the behaviour of rhythmic melody.

[9] Roger Shepard (1929–), American cognitive scientist.

Learning activity

An introduction to the phenomenon of rhythm can be performed very simply: in a quiet environment, tap a finger or a foot at a speed which feels comfortable to you. Measure (with the help of an associate, or by some other means) the frequency of your taps.

Comments on the activity

The majority of people tap at intervals between 200ms and 900ms (frequencies between 1.1Hz and 5Hz). This, along with some other experimental evidence, suggests that there is a preferred temporal interval around 500ms or 600ms.

Musical rhythm, however, is more than just a single beat or tempo; the beats are organised by the listener into groups. Even when the stimulus is a series of equally-spaced pulses (an *isochronous pulse train*, with interval less than about 1s), listeners will spontaneously group them, with groups of two, three or four pulses being overwhelmingly more common than any other grouping. Varying the pulses' pitches, lengths, spacing or amplitudes can encourage one or other grouping to be more favoured.

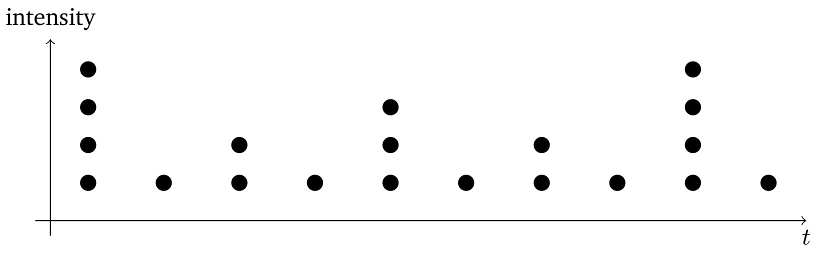


Figure 2.6: A pulse train which is likely to elicit a hierarchical grouping response: groups of two pulses, arranged in groups of two, grouped by pairs once more into a group of eight pulses in all.

Higher groupings can likewise be encouraged; if a pulse train has a strong first element, a slightly less strong fifth element, less strong third and seventh, and weak second, fourth, sixth, and eighth would tend to elicit a grouping response into a hierarchically arranged eight-pulse structure (see Figure 2.6).

2.6 Digital audio formats

Sound is by its nature an analogue (continuously-varying) signal. However, for representation on computer systems, the analogue information must be represented digitally. The following sections describe some details of the audio formats used in current systems, which were introduced in *Creative computing I*, Vol. 2, Section 5.3.

2.6.1 Pulse-code modulation

Pulse-code modulation (PCM) is the simplest form of digital representation of an analogue audio signal; it is used in telephone systems, electronic music keyboards, audio CDs and directly in various audio file formats such as WAVE (commonly in `.wav` files) and AIFF (`.aif` or `.aiff` files).

The process of turning an analogue audio signal into digital form involves two conceptually distinct steps. Firstly, the value of the acoustic signal (the pressure difference) is *quantized* so that it is only possible to represent a limited set of values. Subsequently, the quantized signal is *sampled* at a regular frequency (called the *sampling frequency*), and the value of the quantized signal at that point stored. These two steps are illustrated in Figure 2.7.

The quantization step introduces noise into the signal, distorting the signal's value at any point by up to one quantization step. Sampling the signal at a regular interval reduces the bandwidth, removing information about frequencies higher than half the sampling frequency – the Nyquist frequency, as in *Creative computing II*, Vol. 1, Section 3.3.1 – aliasing them to lower frequencies.

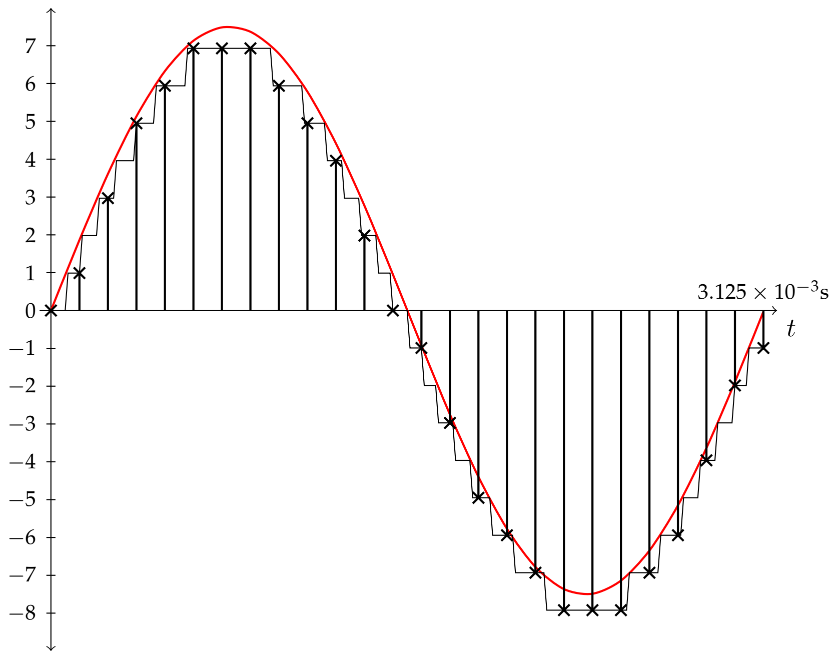


Figure 2.7: Illustration of sampling and quantization of a signal. The sinusoidal signal (thick curve) is quantized to the thin curve, and then sampled at regular intervals indicated by stems and crosses.

In order to reproduce a signal approximating the original one, *demodulation* – the modulation procedure in reverse – is performed. With a time interval of the sampling period, each PCM sample is read in turn, and the system shifted to the new value. These instantaneous transitions will in principle cause the quantization noise referred to earlier, introducing harmonics of the sampling frequency over the original signal; the demodulating system might explicitly filter out frequencies over the maximum resolvable frequency, or else the physical characteristics of the system might damp out these high-frequency harmonics, or they might be at a sufficiently

Technology	**Sampling frequency**	**Quantization level**	**Channel data rate**
Telephones	8kHz	8-bit	64kbit/s
CDs	44.1kHz	16-bit	706kbit/s
DVD videos	48kHz	16-bit	768kbit/s
DVD-audio	192kHz	24-bit	4.6Mbit/s

Table 2.4: Common parameters used in various applications of pulse-code modulation. Some applications will then compress the pulse-code modulated data; see Section 2.6.2 for details.

high frequency as to be inaudible.

Consider the channel data rate for the audio on a DVD video (768kb/s, from Table 2.4). For stereo (two-channel) sound, for a two-hour film, this comes to a total data storage requirement of:

$$768\text{kb/s} \times 7200\text{s} \times 2 \approx 1.4\text{GB},$$

or approximately one-third of the total storage capacity of a DVD (4.7GB).[10] Taking up one-third of a disk just for the audio stream is unreasonable, because the video component is much more data-rich; in fact, the audio component is compressed on the DVD and decompressed by the DVD player. Ways in which digital audio can be compressed are presented in the next section.

2.6.2 Digital audio compression

Perhaps the most straightforward way to compress digital audio would be to apply a generic compression tool such as the `gzip` utility. However, such tools typically compress the audio signal to approximately 70% of its original size, whereas it is possible to achieve significantly better compression by using compression algorithms specialised to the kind of data typically found in audio streams: down to 50% of the original size for lossless compression, and 10%–20% for lossy compression without high signal degradation – see below for an explanation of these terms. `gzip` and related utilities exploit regularities commonly found in text, whereas audio codecs attempt to use regularities in audio signals.

In the field of digital audio compression, the term *codec* is used to mean a format, device or system capable of compressing and decompressing audio; the term may derive from compressor-decompressor or coder-decoder. While pulse-code modulation of an analogue signal, strictly speaking, is an encoding process (and demodulation of the PCM signal to analogue is a decoding process), 'codec' is more typically used in the context of digital audio to mean a system for further treatment (encoding and decoding) of the PCM signal, usually for the purpose of compressing the signal.

[10] The potential for confusion between bits and bytes in abbreviated notation is high; we use 'bit' or 'b' for bits and 'B' for eight-bit bytes.

Lossless compression

It is possible to compress a pulse-code modulated audio signal *losslessly*, without using any details of how humans percieve audio and music, but simply by exploiting redundancy of data. It is always possible to recover the original PCM signal from a losslessly-compressed audio signal.

One example of a lossless audio codec is the free lossless audio codec (or FLAC), a codec with a freely available and usable encoder and decoder.[11] Audio files encoded with FLAC typically have the extension `.flac`. Other lossless codecs exist, such as Apple's lossless audio codec (ALAC) and windows media audio lossless (WMA Lossless) and may be available for your system; we will use details of FLAC's encoding to illustrate some details of lossless compression, but the general principles apply to all of the lossless codecs above.

The design of an audio codec is constrained by certain criteria, of which the most important are adequate fidelity, streamability and compression and decompression performance. For lossless encodings, perfect fidelity (with respect to the underlying PCM data) is assured; streamability is important for the purpose of playing audio from a compressed file without having to decompress it all first.

As for performance, there are tradeoffs. Good compression (meaning a large reduction in the size of the encoded data) will reduce storage requirements or permit more data to be stored on the same device. However, it is important that this does not cause a large increase in the computation required to decode; a typical use for compressed audio files is on mobile music players where an increase in computation requirements will typically both make the device more expensive – by requiring a more powerful processor – and decrease its battery life, by causing the processor to draw more electrical power.

Blocking Streamability of compressed audio is assured by first *blocking* or *framing* the audio signal; each block is then encoded separately, meaning that to play the audio corresponding to each block only that individual block needs to be decoded. While the FLAC format allows for blocks of different sizes, the encoder defaults to a fixed block size for a given PCM sample rate. In the case of 44.1kHz audio, the block size is 4096 samples.

Channel decorrelation One source of data redundancy in audio signals is the connection between the left and right channels in a stereo mix, or between the various channels in a *surround sound* mix: it is likely that the channels are closely related, and so the encoder can attempt to exploit inter-channel correlation. The way to do that is to predict and encode as above not the left and right channels separately, but the difference between them (known as the *side channel*) along with one channel or the average of the two channels.

The FLAC encoder will attempt four encoding modes for stereo sound: left-right, where the two channels are encoded separately; left-side and right-side, where the side channel is encoded along with respectively the left or right channel, and mid-side, where the average and difference are encoded. The encoder then examines the four possible outputs, keeps the best (the most compressible) and records in the encoded stream the choice that has been made.

[11] Available from the project website at `http://flac.sourceforge.net/`.

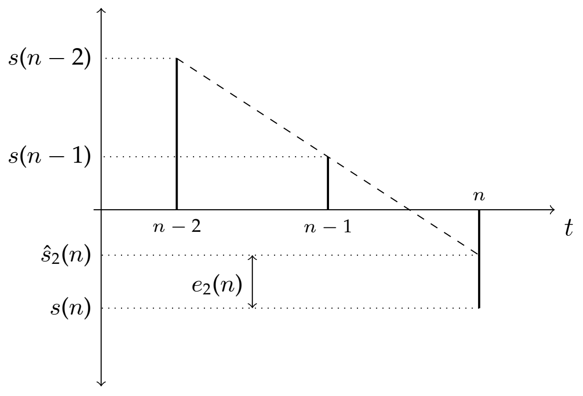


Figure 2.8: An illustration of the process for a line-fitting linear predictor of the form in Equation 2.10. The predicted value of sample n is given by the value of sample $n-1$, added to the difference between samples $n-1$ and $n-2$. Also shown is the residual $e_2(n)$ (see Equation 2.11) between the predicted value $\hat{s}_2(n)$ and the actual value $s(n)$.

Modelling The next step in compressing the audio is to model the signals. The simplest way of doing so is to use *linear prediction* (also known as *linear predictive coding*). The general form of a linear predictor is given by:

$$\hat{s}_p(n) = \sum_{i=1}^{p} a_i s(n-i) \tag{2.8}$$

where $\hat{s}_p(n)$ is the prediction of the value of the signal s at sample n, based on the p previous (observed) samples at $n-1$, $n-2$ ... $n-p$. Specific examples of linear predictors are the first-order predictor, which estimates the value of the signal as being the same as the previous one,

$$\hat{s}_1(n) = s(n-1), \tag{2.9}$$

and the predictor which fits a straight line through the previous two samples to predict the value,

$$\hat{s}_2(n) = 2s(n-1) - s(n-2). \tag{2.10}$$

Figure 2.8 shows the line-fitting predictor, along with the residual error between the predicted value and the actual one.

The FLAC encoder will try the first three orders of these simple polynomial predictors and select the best one, storing the order of the chosen encoder in the compressed stream for the decoder to use.

Residuals The residuals left over from the modelling stage above,

$$e_p(n) = s(n) - \hat{s}_p(n), \tag{2.11}$$

are typically smaller in magnitude than the original signal and are also mostly decorrelated. However, for the original signal to be reproduced exactly from the compressed audio, as is required for lossless encoding, they must be stored somehow. Because of the decorrelation, it is straightforward to use techniques similar to those present in ordinary compressors such as `gzip` to compress the residual data for each frame; details on this aspect will be further explored in the Level 3 module on *Data compression* (CIS325).

Lossy compression

In addition to the compression achieveable by channel decorrelation and signal modelling as in the above section, it is possible to compress an audio signal by removing information (*lossily*, in a way such that the original signal cannot be reconstructed). This need not lead to a perceived unacceptable decrease in the quality of the audio signal delivered from this compression, if the information that was removed was mostly information that would not have been perceived in any case.

There are many lossy compression formats; the most common are MP3 (standing for MPEG-1 audio layer 3, with usual file extension `.mp3`), advanced audio coding (AAC, the default format used by Apple hardware such as the iPod and software such as iTunes; files typically have the extensions `.m4a`, `.mp4` or `.aac`), and Vorbis[12] (a freely implementable lossy codec claimed to be unencumbered by software patents; commonly used within an Ogg *container* with filesystem extension `.ogg`).

The details of how these codecs remove information from the audio stream differ; one component common to all lossy codecs is to remove sounds that are masked according to a *psychoacoustic model*, either in the time domain by temporal masking, or by frequency masking in the frequency domain, reducing the information present in the digital audio while making only limited changes to the perceived sound.

2.7 Summary and learning outcomes

This chapter introduces aspects of sound and music related to auditory perception. The nature of sound as the physical phenomenon of pressure waves is introduced, followed by an examination of how those pressure waves are amplified and converted into electrical signals by the structures in the ear. The relationship between musical pitch of a single sound wave and the frequency of that wave is discussed, as is the harmonic series and its relationship to consonance and harmony; while equal temperament (the division of the octave into twelve equal semitones) is placed in its historical context, and perceptual aspects of melody and rhythm are introduced.

The details of digital audio file formats are covered; pulse-code modulation, as found in uncompressed audio formats, is described in detail; while the methods used to achieve streamable lossless audio compression are illustrated using the free lossless audio codec. Finally, the psychoacoustic model and how it allows lossy compression of audio with minimal loss of perceived quality is discussed.

With a knowledge of the contents of this chapter and its directed reading and activities, you should be able to:

- understand the nature of sound;
- describe how the human ear converts acoustic pressure waves into electrical impulses;
- describe the mechanisms the acoustic system uses to locate the sources of sounds;
- use equal-loudness curves to relate the intensity of stimuli with their perceived effect;

[12] Available from `http://xiph.org/vorbis`.

- discuss how the structure of the inner ear influences the perception of musical pitch and dissonance;
- describe Pythagorean harmony, equal temperament, and how they differ;
- relate perceptions of melody to Gestalt principles of perception;
- describe the hierarchical nature of rhythm;
- describe how to generate a pulse-code modulated representation of an acoustic signal, and common choices of parameters for this operation;
- reproduce the steps to achieve lossless audio compression with the method used in the free lossless audio codec;
- describe the relationship between the psychoacoustic model, masking and lossy compression.

2.8 Exercises

1. Calculate the time difference between the left ear and right ear receiving a signal from a sound source 45° to the left of the listener. Draw a diagram, and state your assumptions.
2. With a Pythagorean scale starting on F (as in Figure 2.5), compute to three significant figures the frequency ratios of the notes of the scale within the octave to F. Arrange the notes in order of their frequency ratios to the starting note.
3. Investigate the quality factor of regions on the basilar membrane, starting from Equation 2.7. Over what regions of frequency, if any, can the Q be treated as approximately constant?
4. Construct a table similar to Table 2.3, for the interval of a minor third (frequency ratio 6:5) and a whole tone (frequency ratio 9:8). Are these intervals more or less dissonant than the perfect fifth and major third?
5. Why is 8kHz an acceptable sampling frequency for Pulse-Code Modulation of telephone signals, whereas 44.1kHz is much more commonly used for sampling music?
6. Examine Figure 2.7, and compute from it the values of the:
   (a) frequency of the sinusoidal analogue signal;
   (b) sample rate;
   (c) quantization level (in bits); and
   (d) data rate of the PCM digital audio.
7. Consider the signal s represented in *Octave* as:
   ```
   octave:> s = 1/3 * (2*sin(2*pi*440*1/44100*[1:256]) + \
                       sin(2*pi*770*1/44100*[1:256]));
   ```
   and compute the residuals from
   (a) the trivial predictor $\hat{s}_0(n) = 0$;
   (b) the slow-motion predictor of Equation 2.9; and
   (c) the line-fitting predictor of Equation 2.10.

   You may wish to refer to the delay system in *Creative computing II*, Vol. 1, Section 4.1.5, or otherwise implement the computation with convolution.
8. Draw a histogram of the residuals for the various predictors from Exercise 7 using *Octave*'s `hist` command. You will need to supply a second argument to put the residual counts into sufficiently fine bins. There will be some extreme values in the residuals from the non-trivial predictors; where do they come from?

Chapter 3

Multimedia information retrieval

3.1 Introduction

The two previous chapters in this subject guide have introduced you to current knowledge and understanding of how we perceive the building blocks of media: light, colour, motion, sound and music.

As well as informing your own creative production, knowledge and understanding of perception is critical to the building of tools for interacting with large databases of media, such as exist on personal devices (such as mp3 players and mobile phones) and at locations on the Internet (such as the flickr photosharing service or the YouTube user-generated video host).

This chapter introduces the concept of multimedia information retrieval, discussing the different tasks that involve finding particular items of media, and how to build components of the systems that perform those tasks. We begin by introducing the fundamental retrieval task.

Additional reading

Witten, I.H., A. Moffat and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. (San Francisco: Morgan Kaufmann Publishing, 1999) [ISBN 1558605703 (hbk)].
Van Rijsbergen, C.J. *Information Retrieval*. Available at `http://www.dcs.gla.ac.uk/Keith/Preface.html`.

3.2 Retrieval

The basic task in multimedia information retrieval is to retrieve one or more pieces of media (often images, music tracks or videos) given some kind of description of the desired media. This description can be extremely specific, for example 'the fourth music track from a certain album', or extremely vague, such as 'a track that I would like to listen to' – there are query tasks at every degree of *specificity*. The description can also be *textual*, as in the examples above, or *content-based*, where instead of using words or a phrase to describe the desired media, another media item is used as the whole or part of the description – or of course, the query description can be a mixture of words and media.

In addition to whether the query is concretely or vaguely specified, there is the issue of whether only *exact* matches are admissible, or whether *approximate* matches (retrieved results which are 'sufficiently similar' in some sense) will be considered correct. The issue of similarity is considered in more depth in Section 3.2.2 below.

3.2.1 Fingerprinting

One multimedia information retrieval task, highly-specific and admitting exact matches only, is to identify an entire item, such as a music track, from a small excerpt. This task is known as *fingerprinting*; the essential idea is to identify unique characteristics (or 'fingerprints') of every database item, and then at query time to compare the fingerprint of the query to each of the database fingerprints.

The simplest solution to this task is to compute a hash value for each of the database items (such as an MD5 or SHA1 hash value), based on the media file representation. This approach fails on two counts, however: firstly, a media item is only retrievable if the item in its entirety is used as the query; secondly, any kind of distortion of the item (including transcoding to a different file format) will lead to a different hash value being computed.

To rectify these defects, the approach taken in real systems is to hash particular characteristics unique to a given item; for example, in the task applied to audio tracks, hashes of localised patterns of peaks in the spectrogram (see Section 3.3.3) are used. This approach brings the dual benefits of being able to match query fragments (as long as the fragment contains at least one of the localised patterns used when constructing the database index), and being resistant not only to transcoding but to many forms of degradation.

The Shazam service[1] works along the lines described above: a query fragment can be transmitted to the retrieval engine over even such a bandwidth-limited channel as a mobile phone and not affect retrieval performance; the fragment can even originate from a noisy environment such as a nightclub (where as well as the music track playing there will be all sorts of ambient noise). However, the fact that exact matching is performed means that there is no scope for retrieval given even very slightly different audio: the retrieval depends on imperceptible artefacts in the sound, so that while a particular recording can be matched exactly, a different mastering (or a live performance) will fail to be retrieved.

3.2.2 Similarity

At first glance, a content-based retrieval task based around *similarity* might seem pleasingly straightforward. Unfortunately, while people often have an intuitive understanding of what similarity means in a particular context, the very fact that it is contextual means that a system to fulfil a similarity request needs to incorporate that context.

Examples of content-based similarity tasks range from the highly-specific, such as finding musical tracks which remix or mashup a query track – where there is clearly a right or a wrong answer – to recommender systems which, for example, suggest tracks that a user might be interested to buy given their current playlist – where there is no correct answer, but the system's performance can be assessed through the quantity of sales made.

This indicates that the notion of similarity is largely task-dependent, or even user-dependent: what is counted as 'similar' material for one retrieval system may be dissimilar for another; two users' similarity judgments (in a retrieval of thematically-related images, for example) may be markedly different.

[1] `http://www.shazam.com/`

In addition, the notion of similarity can be applied to parts of an item as well as to the whole. A *remix* may share only a small amount of musical content with an original track and yet be counted similar. More generally, multimedia information retrieval can be applied both to retrieval of items from a database and to the retrieval of regions of interest within a single media item.

3.3 Features and distance measures

The general description of multimedia information retrieval begins by assuming that *features* of the media items in a database have been computed or are otherwise stored and easily retrievable given a unique identifier for the media item. Examples of such features include: the CIE LAB co-ordinates of the predominant colour of an image; the average loudness of an audio track; the times in seconds of camera angle changes in a video; and the user-supplied tags on a photograph.

These values, which can either be numerical or textual, can be thought of as existing within a *feature space*, which is a way of describing all possible values of a feature as well as the way that those features can be compared. For example, as discussed in Section 1.4.4 of this subject guide, the CIE LAB space is a feature space where the colour co-ordinate values are real numbers, and the ordinary distance between those co-ordinate values corresponds to the perceived distances between the corresponding colours: given a reference colour, colours perceived to be more similar to the reference will have a smaller ΔC in CIE LAB space than those percieved to be less similar.

3.3.1 Textual features

Although this chapter is about multimedia information retrieval, textual features remain important for two reasons. Firstly, many media items are associated with textual *metadata*: information about what the item is, who created it, when it was created. Secondly, it has recently become popular on media websites with a social component, such as flickr[2] or last.fm[3] to allow *collaborative filtering* or *tagging* of media items, associating a set of words with that media item based on the contribution of many users.

These two kinds of textual features then allow for retrieval based on them. In the first case, it is possible to retrieve a media item based for example on its title and/or on the name of its creator. In the second case, media items considered to have sufficiently similar sets of tags can be associated, which potentially allows the retrieval of a group of similar media based on the labelling activities of many users.

In addition, it is possible to associate media items with documents discussing them, and then retrieve those media items if the documents match search terms sufficiently closely. This technique of *text mining* is beyond the scope of this subject, but is the essential strategy behind tools such as Google's image search.

In all kinds of textual information retrieval, certain text-manipulation techniques are used. Firstly, *stopwords* or *noise words* – overwhelmingly common words such as 'the', 'and', 'you' and 'who' – are removed from consideration, so that their presence

[2] `http://flickr.com/`
[3] `http://www.last.fm/`

does not cause uninteresting retrieval simply because both query and retrieved document contain these stopwords.

Secondly, *stemming* might be performed, in order to attempt to normalise inflected or derived words to the same form: plurals, verb participles and the like all being converted to the same internal word, so that queries for 'query' will match documents containing the word 'queries', and searches for 'sleepily' might retrieve documents containing 'sleep'.

Learning activity

Investigate the function and history of the *Porter stemmer*, due to Martin Porter. Find a rights-free program implementing Porter's stemming algorithm (for example from `http://tartarus.org/~martin/PorterStemmer/`) and experiment to find what it does to words of your choice.

The use of a set of user-supplied tags as a textual feature has led to the construction of so-called 'folksonomies': bottom-up classification systems formed when considering social tags. Despite the name, a folksonomy does not have much in common with a *taxonomy*, which is a top-down hierarchical classification system (such as is found in libraries, for example); instead, the presence or absence of each tag is itself a categorisation decision. Folksonomies are an emergent property of large-user interactive online communities, often given the label 'Web 2.0'.

Distance measures for textual features

The simplest distance measure for a textual feature is a binary one: does a query string match a target exactly or not? This distance measure corresponds to exact string matching, discussed in *Software engineering, algorithm design and analysis*, Vol. 2, Section 9.5. However, it suffers from being over-specific, which is why techniques such as stemming were developed.

Even given a good stemming algorithm (see the previous section), it is possible for a search term not to be stemmed to a word present in the document database: for example, when a word is misspelt. Under such circumstances, it is perhaps useful to compare words or, more generally, text strings, and compute a usable distance between them. The distance used for that purpose is known as the *edit distance*, which in fact is a parameterised family of distances.

In its most common form, the edit distance (in this context also known as the *Levenshtein distance*[4]) is the minimum number of operations required to turn one string into another, where the operations are: insertion of a character; deletion of a character; and substitution of a single character with another.

For example, the Levenshtein distance between the words 'turn' and 'fun' is **two**: one to convert the initial 't' of 'turn' into an 'f', and one to delete the 'r'. Algorithms implementing the computation of the Levenshtein distance commonly use the technique known as *dynamic programming* (see *Software engineering, algorithm*

[4] Vladimir Levenshtein (1935–), Russian mathematician.

design and analysis, Vol. 2, Section 3.5), which has many applications in matching sequences.

Learning activity

The name of the Russian composer Пётр Ильич Чайковский is commonly transliterated into English as ‘Piotr Ilyich Tchaikovsky’. Try to find alternative transliterations, and compute edit distances between these transliterations. Verify that the edit distances from ‘Modest Petrovich Mussorgsky’ to the transliterations that you have found are larger than the edit distances amongst the transliterations.

Comments on the activity

While ‘Tchaikovsky’ is the most common transliteration of the surname into English, the same is not true in other languages; for example, the most common transliteration in French is ‘Tchaïkovski’, while in Spanish it is ‘Chaikovski’. The edit distance is a way to try to mitigate against differences in transliteration, as well as to correct spelling errors in a user’s query.

A special case of the general edit distance is the *Hamming distance,*[5] where the distance between two strings of the same length is the edit distance where the only permitted operation is substitution. The Hamming distance therefore computes the number of places where the strings differ.

This distance can be used to compare tags on media items. If the global vocabulary of tags is ordered, alphabetically for example, then the tags applied to a particular media item can be represented by a bitstring, where a 1 in the string corresponds to that tag being present and a 0 when the tag is not present. The distance between two media items, based on their tags, can then be taken as the Hamming distance between the two bitstrings. Of course, this distance measure over those tags ignores possible semantic associations between tags, treating differences in tag complements equally, whatever the tag.

If the text search desired is more like matching search terms against documents, in a similar way to a Web search engine such as Google’s, then a more appropriate measure of appropriateness is the term-frequency–inverse-document-frequency measure.

The *term-frequency* component measures, for each document, the number of times that term (which may be a stemmed version of a query word) is present in the document. The more times a document contains a term, the more likely it is to be relevant for queries for that term. In practice, rather than the absolute count of mentions of a term, the proportion of a document taken up by a term is used:

$$\mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} \tag{3.1}$$

where the number of times term i occurs in document d_j is given by n_{ij}.

However, a common term will be contained in many documents. In order to maximize the importance of the terms that distinguish relevant from non-relevant retrieval results, the term-frequency is weighted by the *inverse-document-frequency*,

[5] Richard Hamming (1915–1998), American mathematician.

measuring the number of documents in which the term is present at all:

$$\mathrm{idf}_i = \log \frac{|D|}{|d_j : n_{ij} > 0|} \tag{3.2}$$

where $|D|$ is the number of documents in the database.

Combining these two by multiplying them, a high tf-idf value for a term corresponds to a document containing many mentions of a term in a database of documents where few refer to that term at all.

Applications of textual features

Textual features are used in many different online multimedia applications. Google image search,[6] for example, works by associating images in web pages with linking text and other textual material close to the image, under the assumption that that text is likely to be relevant to the image. This is a typical text-mining application.

By contrast, the collaboratively-built online music metadatabase Musicbrainz[7] allows users to upload metadata of their music collections, and applies careful oversight of the metadata it contains to ensure consistency. The Musicbrainz website allows text search of various metadata fields such as 'Artist', 'Track' and 'Release'.

3.3.2 Numerical features

Distance measures for numerical features

Suppose that we have two numerical features, X and Y, in an N-dimensional feature space, so that the co-ordinate representation of X is $\{x_1, x_2, ..., x_N\}$ and the representation of Y is $\{y_1, y_2, ..., y_N\}$. The *Euclidean distance*,[8] which corresponds to the usual meaning of distance (such as the distance along the hypotenuse of a triangle), between p and q is given by:

$$\begin{aligned}\Delta^{(2)}_{XY} &= \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_N - y_N)^2} \\ &= \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}. \end{aligned} \tag{3.3}$$

Figure 3.1 illustrates this distance measure for a three-dimensional feature space.

The Euclidean distance is also known as the *Pythagorean distance* or the 2-norm distance.

While the Euclidean distance is the most commonly-used distance measure in feature spaces, probably because of the correspondence with distance in real space, it is not the only distance measure. The Manhattan distance (also known as the 'city-block' or 1-norm distance) is sometimes useful when the meanings of the different feature co-ordinates are distinct; it is given by:

$$\Delta^{(1)}_{XY} = |x_1 - y_1| + |x_2 - y_2| + ... + |x_N - y_N|$$

[6] `http://images.google.com/`

[7] `http://musicbrainz.org/`

[8] Euclid of Alexandria (fl. 300BC), Greek mathematician and 'father of geometry'.

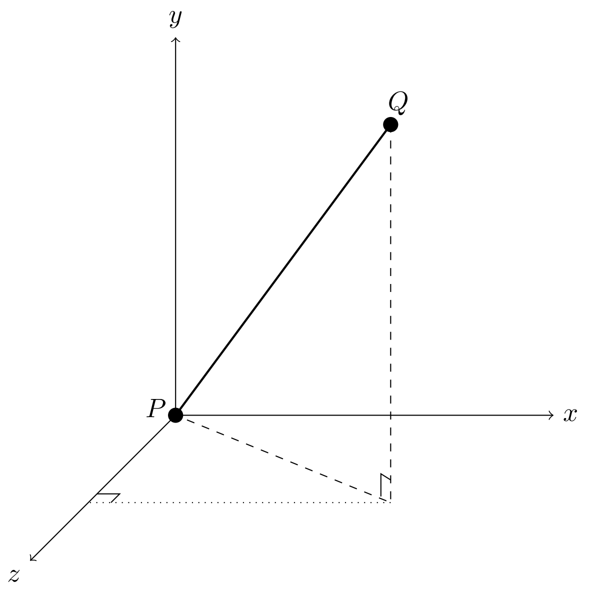


Figure 3.1: Projection illustrating the Euclidean distance between points P and Q in three dimensions. The hypotenuse of the right-angled triangle in the $x-z$ plane forms one of the shorter sides in the right-angled triangle with the overall Euclidean distance as its hypotenuse.

$$= \sum_{i=1}^{N} |x_i - y_i|. \tag{3.4}$$

The two distance measures above generalise to a p-norm distance: where the Euclidean distance takes the square root of the sum of squared co-ordinate differences, and the Manhattan distance takes the unit root of the sum of co-ordinate differences, the p-norm distance takes the pth root of the sum of co-ordinate differences raised to the power p:

$$\begin{aligned} \Delta_{XY}^{(p)} &= \sqrt[p]{|x_1 - y_1|^p + |x_2 - y_2|^p + \ldots + |x_N - y_N|^p} \\ &= \sqrt[p]{\sum_{i=1}^{N} |x_i - y_i|^p}. \end{aligned} \tag{3.5}$$

The generalisation to p-norm distances is only valid for numbers p greater than or equal to 1, although non-integers are allowed. There is little practical use for any p-norm distance for p not equal to one or two, apart from the special case of the norm with $p = \infty$, also called the Chebyshev distance,[9] which is given by the maximum co-ordinate distance:

$$\begin{aligned} \Delta_{XY}^{(\infty)} &= \lim_{p\to\infty} \sqrt[p]{\sum_{i=1}^{N} |x_i - y_i|^p} \\ &= \max_i |x_i - y_i|. \end{aligned} \tag{3.6}$$

[9] Pafnuty Chebyshev (1821–1894), Russian mathematician.

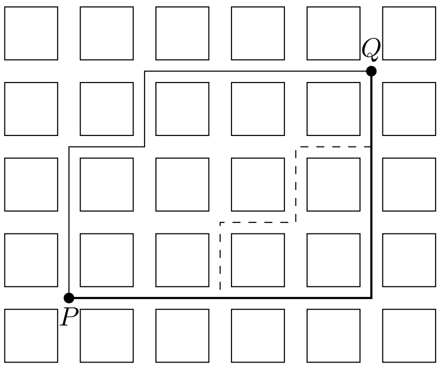


Figure 3.2: An illustration of the Manhattan distance for a two-dimensional space: each of the paths between P and Q, depicted by the thick, regular and dashed lines, has the same length, in this case seven 'city-block' units.

For scalar features, all p-norm distances are equivalent: they all reduce to the absolute difference between the two scalar values. It is for vector features that the exponent p makes a difference.

There are other distance measures that are occasionally used in multimedia information retrieval: some correspond to particular geometric setups, such as a distance measure given by the angle between two points on a sphere; while others are motivated by a particular interpretation of the features. When a feature corresponds to a probability distribution, for example, the Kullback-Liebler divergence, which measures how different two probability distributions are, might be appropriate; for discrete probability distributions P and Q (having outcomes i with probabilities p_i, q_i such that $\sum_i p_i = \sum_i q_i = 1$), the Kullback-Liebler divergence is given by:

$$\Delta^{(\mathrm{KL})}_{P||Q} = \sum_i p_i \log \frac{p_i}{q_i} \tag{3.7}$$

Learning activity

Investigate the properties of the Kullback-Liebler divergence. Why do you think it is not called the Kullback-Liebler 'distance'?

Comments on the activity

The Kullback-Liebler divergence has some slightly odd properties. Firstly, the KL divergence between distributions P and Q is not the same as the divergence between Q and P. There is a modified divergence known as the Jensen-Shannon divergence,

$$\Delta^{(\mathrm{JS})}_{PQ} = \frac{1}{2}\left(\Delta^{(\mathrm{KL})}_{P||Q} + \Delta^{(\mathrm{KL})}_{Q||P}\right)$$

which makes the divergence values between Q and P equal. However, even with this modification, the second odd property of the KL divergence remains: the *triangle inequality* is violated. The triangle inequality states that the distance between points A and C, Δ_{AC} is always less than or equal to the sum of the distances Δ_{AB} and Δ_{BC} for any point B. This is a desireable property for distances to have, because it allows the construction of efficient algorithms. Since the p-norm distances for $p < 1$ also do not satisfy the triangle inequality, these distances are rarely used; the Kullback-Liebler divergence is used in information retrieval despite its violation of the triangle inequality.

3.3.3 Examples of perceptual features

This section covers examples of perceptual features that can be used in building systems for multimedia information retrieval. The features that are used in any given system depend on the task that is ideally being performed, as well as potentially the details of the collection being retrieved from. Numerical features can be *scalar* features (a single number) or *vector* features (a fixed number of numerical entries).

Vision

A simple, scalar feature for still images is the overall intensity or *luminance*. As discussed in Section 1.4.4, a perceptual correlate of intensity such that equal perceptual distances correspond to equal Euclidean distances in space is given by

$$L^* = 116 f\left(\frac{Y}{Y_0}\right) - 16 \tag{3.8}$$

where the function f is defined in Equation 1.9, Y is the CIE XYZ co-ordinate of the light in question and Y_0 the Y co-ordinate of the reference colour.

This feature will allow retrieval of images of similar average intensity from a collection, and indeed to rank a collection in terms of perceived difference from a query image's intensity. Usually, however, a more detailed retrieval is desired: for instance full-colour matching rather than simply the intensity. The CIE $L^*a^*b^*$ space, defined in full in Equation 1.8 has been designed in order to match the Euclidean distances between the CIE co-ordinates of two colours, and the perceptual differences reported by experimental subjects; the CIE $L^*a^*b^*$ co-ordinates make up a vector feature for colour.

Learning activity

The 'flickr colr pickr' application, available online at `http://krazydad.com/colrpickr/`, uses images from the 'colour field' group at flickr to build a colour chooser, but using photographs instead of solid colours in its user interface. Experiment with using it; what does it add over a regular colour chooser widget?

Download some of the colour field pictures, and compute their average CIE LAB co-ordinates. Try to build a small program that displays the closest image to a user-specified colour.

Comments on the activity

Whether or not the implementation of the colour chooser available online uses CIE LAB co-ordinates for its distance measure does not matter too much; for retrieval from a large collection of images, where the nearest match in the database will be small, small sRGB distances will correspond approximately to small perceptual distances, and in the flickr colr pickr only a handful of results (with small distances) are retrieved.

Proximity in colour space can be used to create composite images through the technique of *photomosaicking*: the colour of each pixel in a source image is used as a query against a database of other images: then a larger image which will resemble the source image (through colour mixing by averaging, as discussed in Section 1.4.3) can be constructed by placing the retrieved images in positions corresponding to their respective query pixels.

Features corresponding to geometrical patterns within images are used in *face detection*, present in digital cameras and in other photographic applications (such as tools to remove red-eye effects from flash photography). The general technique is to build a system (in the sense of *Creative computing II*, Vol. 1, Chapter 4) whose response is strong when a portion of an image signal contains arrangements resembling eyes, nose and mouth; the technique is complicated by having to recognise faces at different rotations and scales.

Animation

In tools for video editing, it is useful to be able to determine automatically when the camera angle has changed via *shot detection*. This allows the *segmentation* of a video into distinct pieces, each corresponding to a particular shot.

One way to approach this is to consider *difference features*: the difference between feature values in successive frames. The intuition is that strong differences in intensity, colour or other such image features between two successive frames are likely to be caused by the frames being part of two different shots.

However, picking out all local maxima in difference features tends to cause *oversegmentation* or *fragmentation*, as natural random variation in the signal causes spurious shot changes to be detected. To deal with this, either or both of *thresholding* (where only sufficiently large – above a threshold – maxima in the difference feature values are considered for shot changes) and imposing a minimum segment length can reduce the fragmentation.

Sound and music

General audio features include the power or energy present in the signal, and the overall spectrum. The power in a given time window of the audio signal is a single scalar feature. However, using the raw value of the power in perceptual measures of proximity is not appropriate, because (as discussed in Section 2.2.4) the perception of loudness is approximately logarithmic. Therefore, the power should be expressed on the decibel scale (or some other logarithmic scale) before taking differences.

As with colour, a single scalar feature describing the overall intensity of the signal is not strongly discriminatory, and signals with very little overall similarity will be retrieved together. For features which capture more of the fine detail in sound, we use the *spectrum* (as introduced in *Creative computing II*, Vol. 1, Chapter 4) of the signal, as well as features derived from certain transformations of the signal.

The signal spectrum over a window of fixed length in time, as produced for example with the *Octave* command `fft`, can be treated as a vector feature, often known as the *spectrogram*. However, the resulting features are not directly mappable to simple perceptual qualities, for two reasons. Firstly, the spectrum produced by `fft` is binned linearly in frequency, whereas human pitch perception is logarithmic in nature: a musical interval is determined by the ratio of frequencies, not the difference between them. Secondly, the spectrum contains information not only about the amplitude of the component waveforms but also their phase, and while it is true that the auditory system makes use of phase information (as in sound source location, Section 2.2.3) it is not a straightforward mapping.

What is typically done, instead of using the spectrum directly, is to compute the *power spectrum* (the squared magnitude of each spectrum entry), and then collect the power into logarithmically-spaced bins. In this way we have removed the phase information (by taking the squared magnitude) and aggregated the effect of nearby frequencies in a perceptually reasonable way.

When analysing Western music, it is common to choose to arrange the logarithmically-spaced bins such that there are 12 bins within each doubling of frequency (each octave), as that corresponds directly to the system of pitch prevalent in Western music. In addition, octave equivalence (discussed in Section 2.3.2) suggests that a useful additional transformation would be to add together the contributions from all the logarithmic bins separated by octaves, forming a vector feature called a *chromagram*, which represents the musical pitch content of an analysis window independent of octave.

While the chromagram is a feature measuring pitch content, a different transformation of the logarithmically-banded power spectrum yields a different vector feature representing timbre: the *cepstrum*, formed by taking the real part of the `fft` of the logarithmically-banded spectrum.

Both the chromagram and the cepstrum are used in multimedia content-based retrieval systems, in combination with a variety of distance measures, including the p-norm distances.

As well as retrieving entire tracks based on content similarity, audio features can be used to power a music performance tool to perform *musaicing* (a term formed by analogy with 'mosaicing' to describe the construction of a piece of music out of segments of other pieces): for instance, producing a track that approximates *Hey Jude* using only fragments of an orchestral recording of Beethoven's Symphony No. 5.

There are also musical features away from the acoustic domain; the simplest example is probably the *Parsons code*, which is essentially a scalar difference feature for melodies: for each note apart from the first in a melody, the Parsons code expresses whether that note is pitched higher, lower or the same as the previous note. A search engine for musical melodies, Musipedia[10], allows users to search for such *melodic contours*: a search for '`*rururd`' (where '`u`' means 'up', '`d`' means 'down' and '`r`' means 'repeat') retrieves, among other melodies, the French folk-tune *Ah! vous direz-je maman*, known in English as *Twinkle, twinkle little star*.

[10] `http://www.musipedia.org/`

3.4 Systems for multimedia information retrieval

3.4.1 False positives and false negatives

There are two distinct ways in which a retrieval system can fail to behave in a desired manner: it can retrieve items which are not desired by the user (*false positives*); and it can fail to retrieve items which are desired (*false negatives*). When designing a system, it is often worth considering which kind of failure is worse. For example, in a multimillion-document search engine (such as Google's Web search engine), false negatives are not too critical, because there are usually enough *true positives* to satisfy the user, while a highly-ranked false positive is bad, as it will decrease the perceived quality of the search engine. By contrast, in a system where all the items are known to the user, and the user is using the system to retrieve a specific item, a false positive is significantly worse.

The performance of information retrieval systems is characterised by two quantities, *precision* and *recall*. Precision measures what proportion of the retrieved items were in fact relevant to the query (the proportion of true positives to all positives, true or false):

$$\begin{aligned} P &= \frac{|\{\text{relevant items}\} \cap \{\text{items retrieved}\}|}{|\{\text{items retrieved}\}|}; \\ &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}. \end{aligned} \tag{3.9}$$

Recall measures what proportion of the relevant items in the database have been retrieved by the system (the ratio of true positives to all relevant items):

$$\begin{aligned} R &= \frac{|\{\text{relevant items}\} \cap \{\text{items retrieved}\}|}{|\{\text{relevant items}\}|}; \\ &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}. \end{aligned} \tag{3.10}$$

These two measures are linked, in that increasing the number of retrieved documents tends to increase recall and decrease precision. It is usual to state the value of precision at a certain level of recall, for instance by asserting that a system has '60% precision at a recall rate of 75%'.

3.4.2 Small collections

Systems for multimedia information retrieval over small collections of items can be built relatively simply, using exhaustive search over linear data structures.

The database of items is represented as a data structure such as an array (see *Software engineering, algorithm design and analysis*, Vol. 2, Section 2.6) or a linked list (see *Software engineering, algorithm design and analysis*, Vol. 2, Section 2.7), where each entry in the data structure contains a reference to the media item in question, along with the corresponding feature values. A content-based query can then be performed by first performing *feature extraction* on the query, extracting the same features as are in the data structure. Then the distance between the query features and each of the database entry features is computed in turn, and the entries corresponding to the k smallest distances returned as the retrieval results, where k is the number of desired results.

For further information about the details of traversal and searching through data structures, see *Software engineering, algorithm design and analysis*, Vol. 2, Chapter 5.

3.4.3 Large collections

The problem with the exhaustive search strategy of Section 3.4.2 is that it necessarily scales linearly with collection size: the time for near-neighbour search is $O(N)$ where N is the number of items in the database. While this works for small collections, there comes a point when the time necessary to perform the exhaustive search exceeds some limit; whether that limit is a technical one such as a network timeout or a more practical issue such as the user's attention span.

Learning activity

Take your implementation of a linear, exhaustive search for a one-dimensional feature, and estimate how many media items there could be in a collection while your implementation takes less than 10 seconds to complete its exhaustive search.

Given such a collection, consider what the effects would be of:

1. parallelising the search over a cluster (of say 100) machines;
2. implementing a binary search tree data structure for that collection.

The scale at which a collection can be considered 'large' depends also on the level of detail: if each document is considered as a single entity, and *document-level retrieval* is being performed, then less work needs to be done than if the system is designed to retrieve particular pieces of documents, or perform *passage-level retrieval*. Search engines for the Web, such as Google's or Yahoo!'s, typically index billions of web pages, and definitely fall in the large category; an exhaustive search over such a database is completely inappropriate.

There are two strategies for dealing with larger collections. One is to exploit a common feature of searching through these collections, which is that the problem is usually completely paralleliseable: if instead of having a single computer to hand, one has a cluster of one hundred computers, then a typical retrieval can be done in one hundredth of the time, simply by giving each computer one hundredth of the database to traverse and combining the results at the end.

However, exploiting the so-called *embarassingly parallel* nature of retrieval only goes so far, because purchasing ever-larger clusters of computers – not to mention the electricity to power them and the air-conditioning to cool them – rapidly becomes expensive. Therefore, better techniques for searching are required.

The second strategy is to build a data structure to allow for more efficient searches than the exhaustive search, by making it possible to find items in a region around a query point without having to iterate through the entire database. For scalar features, you have already encountered a suitable data structure: the binary search tree, in *Software engineering, algorithm design and analysis*, Vol. 2, Section 5.7. The binary search tree allows the search strategy to rapidly find the vicinity of the query and attempt to satisfy the retrieval request using only a subset of the database, which alters the average time complexity of the search from $O(N)$ to $O(\log N)$.

Unfortunately, the binary search tree only works for one-dimensional (scalar) features; for vector features the situation is more complicated. For low dimensions (up to about four or five), *spatial tree* analogues of the binary search tree can be devised with similar properties, making the average search time logarithmic in the size of the database. Higher-dimensional features, because of the *curse of dimensionality*, make the tree strategy degenerate to linear search, and require probabilistic indexing techniques such as *locality-sensitive hashing*, which is beyond the scope of this subject.

3.5 Summary and learning outcomes

This chapter introduces the concept of information retrieval, and raises issues particular to the retrieval of information in the context of multimedia. A brief introduction to textual features is provided, but while textual metadata and features play a part in multimedia information retrieval, so too do content-based features. Scalar and vector numerical content-based features and distance measures are discussed, including examples from video, audio and still image media items. All of these things are then combined to provide the outline of a complete multimedia information retrieval system, along with a discussion of the issue of scalability.

With a knowledge of the contents of this chapter and its directed reading and activities, you should be able to:

- describe the fundamental task of retrieval;
- describe the concepts of specificity and similarity;
- discuss simple uses of textual features in the context of multimedia retrieval;
- select an appropriate textual distance measure for particular tasks;
- compute the Euclidean, Manhattan, Chebyshev and p-norm distances, and the Kullback-Liebler divergence;
- identify suitable scalar and vector content-based features for image and sound retrieval tasks;
- describe the use of difference features in shot detection for video;
- characterize an information retrieval system's performance in terms of precision and recall;
- understand the linear search model for information retrieval systems, and why it is inappropriate for large databases.

3.6 Exercises

1. As mentioned in the text, textual data is often processed to remove stopwords. Suggest a reason why the removal of stopwords is not always a good idea, giving an example of a situation where it will have an undesired effect.
2. Compute the Hamming distance and the Levenshtein distance between the strings '`The quick brown fox`' and '`Thequick brow n fox`'. Hence give a situation where using the Levenshtein distance between strings might be more appropriate than using the Hamming distance.
3. For points P, Q and R at co-ordinates $(1.3, 2.6, 1.4)$, $(1.7, 1.2, 2.8)$ and $(2.7, 1.3, 1.8)$ respectively, compute:
   (a) the Euclidean distances PQ and PR;
   (b) the Manhattan distances PQ and PR; and
   (c) the Chebyshev distances PQ and PR.
4. Two pieces of music, in the keys of C major and C minor, have corresponding 12-dimensional chromagram features given by:

C major	$\{5, 1, 3, 1, 4, 2, 2, 6, 2, 2, 2, 3\}$
C minor	$\{5, 2, 3, 4, 1, 2, 1, 6, 3, 2, 2, 2\}$

   Which of these is closer to an A minor query chromagram, with feature values $\{4, 1, 2, 1, 6, 3, 2, 2, 2, 5, 2, 3\}$, under the Manhattan distance? If you can, relate this to Western music theory.
5. An information retrieval system with 1000 items contains 50 relevant items for a particular query.
   (a) The user first uses the system to retrieve 10 items, of which 9 are in fact relevant and one is not: compute the recall and precision for this retrieval.
   (b) The user then expands their search to 50 items; the system returns 45 relevant items and 5 non-relevant ones. Compute again the recall and precision for this retrieval.

Chapter 4

Animation

4.1 Introduction

To animate something means to 'bring it to life'. In this chapter we look at some of the basics of animation, in the context of creativity and computation.

4.1.1 Some background

The history of animation starts long before film and computers. As a very simple and low-tech example – and one you may already have done yourself as a child – consider taking a deep, small and sturdy pad of paper. On the bottom-most page, draw any image. A clear and simple one is best. Draw another on the second-last page. Then on each consecutive sheet in the pad, draw a new version of the image, which is only slightly changed from the previous page. When you draw the images, make the changes happen to the same general part of the image. Now, riffle through the pad using your thumb, from the back to the front, in much the same way as you might riffle a deck of cards. Watch the image when you do this: you have created your first animation.

Flip books, or flick books, were early forms of animation, created long before film was invented. There are examples of children's flip books from Victorian times. Some modern examples of flip books can be found on the Internet; there are many more than those listed here.

- `http://www.youtube.com/watch?v=xSrDnIVgVvO&NR=1`
- `http://www.youtube.com/watch?v=WQGDO4hs76g`
- `http://www.skype.com/allfeatures/hello/flickbookstory/`

Learning activity

Make a flip book.

Media animation works in much the same way as described above – to give the impression of life, we create something that appears to move over time. This is done by having the position of either the object itself, or a part of the object, change, in a way similar to how something would move in real life.

So, animation in the artistic sense relies on the creation of an optical illusion. There are a few different approaches to obtaining the effect of animation; all have in common the use of extremely small differences in image, to give the impression over time that something is moving. This is something that relates directly to perception, which has been discussed in detail in Chapters 1 and 2.

In general, we think of animation in terms of moving images, and visual artistic expression, but another very important use of animation is in the conveying of information and messages. This is broadly called *visualisation*, and is discussed more in Section 4.5. Now we look at the two main approaches that have been used for creating an animated film or video, and then look at computer animation.

4.2 Approaches to animation

Prior to the use of digital technology, there have been two main ways to make animated films. We can take a series of drawn images that differ from each other only very slightly, and combine them into a film, using a similar approach to that described above. This kind of animation is called *flat animation* or traditional animation, and is how the first animated cartoons were made.

A second approach is to use models, usually made of clay, and repeatedly take a snapshot or photograph of them, then move them or alter them a very small amount, and take another snapshot. *Stop-motion animation* is produced in this way.

How animations are made depends heavily on the type of animation.

Flat animations Traditionally, flat animations were a sequence of drawn images. This means that each frame had to be drawn separately by hand. This could take an enormous amount of animators' time, and the large studios developed many methods to deal with this problem. For example, characters were traditionally drawn in a fairly simple style to minimise the amount of work needed, while backgrounds (which did not have to be re-drawn for each frame) could be highly detailed. Animations were broken down into set sequences that could be repeated, for example 'walk cycles'. Finally, one of the major problems with the animation workload was that it was difficult to distribute to different people. Thus, if many animators worked on an animation sequence then the resulting style would be inconsistent.

To solve this problem the technique of *keyframing* (see Section 4.4.3 below) was invented. A single lead animator was responsible for creating the animation including its style. However, he or she only needed to draw as many frames as were needed to define the movement in the animation (the key frames). While these frames are sufficient to show what the animation will look like many more frames are needed to make the movement look smooth (the in between frames). A team of assistant animators performed the more menial job of drawing these in between frames ('inbetweening' or 'tweening').

Stop-motion animations Stop-motion animations are created very differently. Rather than a sequence of images, characters are solid models, made of clay or a similar material. Generally one model is used per character (or perhaps one for long shots and one for close ups) and this model is moved in each frame to produce the animation. This results in a very different work flow. A large amount of effort is put into initially creating the character model, to ensure that it both looks good and can move effectively. Once this is done, the actual work of animation consists of manipulating and moving the model.

Computer animation Computer animation, whether flat 2D or 3D, is generally much closer to the stop-motion work flow. A model of each character is initially created, but this time it is a virtual model in 2D or 3D. The appearance of the model has to be defined before starting the animation, but so do a number of manipulation handles that are used to move the character. These handles are called the 'rig'. Once the model and rig are defined the animator's job is to move the model frame by frame, as in stop motion animation. However, the method of *keyframing* has been borrowed from flat animation. The animator defines a set of key frames, but it is the computer, not a team of animators, that automatically performs the inbetweening.

Learning activity

What examples of films can you find that contain animation?

Find five examples, and for each, describe how you believe the animation has been created.

There are advantages and disadvantages to each of the above approaches. Discuss what the limitations of each approach might be.

Learning activity

An early animation from Pixar studios is called '*Pixar light and heavy*'.[1] Find it and look at it, and think about the following things in the context of the short piece. If you have colleagues who are interested in this work, you can discuss it with them, or you could write a critical evaluation of the work.

- What makes it lifelike?
- What kind of animation is it?
- What are the factors that contribute to its success? Can you think of differences that would make it work less effectively?
- Could you add anything to the animation that would make it even better? How would you go about doing this?

Pixar have a number of other short animated films, that come from their *Sesame Street* series. Find as many of the others as you can on the Internet, and evaluate them in a similar way.

Based on the above, come up with an idea for a new short animated film. Describe the main plot, and the visual hooks that you would use. Try to create a storyboard for the animation, as one of the first steps to production.

---

[1] Available at `http://www.youtube.com/watch?v=mjLvLytm45w`.

4.3 Perception in video and film

The main difference between conventional (acted) video or film, and animated video or film, is that in animation, the images themselves are either created by artists, or are snapshots of objects created by artists. Apart from this, there are many similarities, and so a brief look at how we see movement in film is informative.

Even though film was invented decades before the digital age, there is an aspect of film that is discrete, rather than continuous. When we watch a film, it appears as though there is a continuous image stream; however, what we are actually seeing is rapid presentation of slightly differing images, much like that of a cartoon.

As discussed in Section 1.5.2, the *frame rate* is the rate at which the presentation of these images occurs; each image is a *frame*. If the frames are presented to the viewer in a rapid enough sequence, the viewer is not aware of the discrete nature of the presentation. In the earliest movies, frame rates were around 10 frames per second (fps). 16 fps is the minimum at which no jitter is obvious, and current video formats use between 24 and 60 fps.

It is not simply the case that the tradeoff is between frame rate (where a higher frame rate implies greater storage space needs) and quality. The actual techniques of presenting the images to the viewer also play a part in the result. Images can be *interlaced*, for example, and the way in which an image gets replaced by the next one depends on the technology being used. Video and film differ markedly in this, and the digital aspect also has an effect.

Learning activity

Find out what the difference is between 'interlaced' and 'progressive' scanning.

What is 'flicker'?

How do both of these concepts (scanning and flicker) relate to digital video?

4.4 Making animations

Animation has moved from something that was done in film studios to something that individuals can do on their home computers. While it is still the case that animation is a big part of the work of some major studios, many new advances in both technological and artistic domains are being made by individuals working at home, and research workers in university and research labs. This move has been due to the advances in digital technology that have made computing power and software tools available at a fraction of the cost that they used to be.

4.4.1 What is involved in making an animation?

Animations are made in a number of different ways. An animation can last for five seconds, and have a huge impact. On the other hand, there are full-length animated feature films. For all of these, there are some basic commonalities: a creative idea, that relies on movement to get its message over; a story (in the sense that 'something happens', or a number of related things 'happen'); a realisation of the story (however simple or complex) into some kind of film or video or digital medium.

Learning activity

There are a number of phases in the creation of an animation, from storyboarding and design, to final execution. What do the different phases involve? Come up with a list that covers all of the necessary phases.

It is a useful exercise to look at the different phases you have identified, and to discuss the appropriateness of digital or non-digital mechanisms for implementing them.

4.4.2 Animation tools

There is a variety of tools that assist in the creation of animation, from tools that help with the drawing of a single frame, to tools that can be used to create an entire animation.

Much of what is done in modern animation uses digital technology. However, there are still some parts of the entire animation process that do not. Storyboarding is still usually done on paper. In many studios, the production of images is a mix of hand-drawing and subsequent refinement using a computer tool.

Adobe's *Flash Animation* suite can be used to create a digital animation. At the other end of the scale, *Processing* can be used to create a digital animation. With Flash, what you would be doing as the animator is providing high-level instructions as to how you'd like the animation to behave; with *Processing*, you manipulate at the pixel level, to provide the perception of movement. It is important to understand the differences in these ways of animating, which are at opposite ends of the spectrum.

Have a look at the Shel Silverstein website[2] for an example of the use and range of digital animation.

4.4.3 Keyframing

Another way of carving up the computer animation space is to consider two kinds of approaches: keyframe animation and simulation. In *Creative computing I*, Vol. 1, Chapters 8 and 9, you looked at examples of the latter (particle systems, artificial life, Brownian motion, Perlin noise, using *Processing*). These approaches use algorithms to create animation, or something 'lifelike'. In this chapter we focus on keyframe animation, which is data driven.

[2] `http://www.shelsilverstein.com/`

As mentioned earlier, animation is effectively a series of images (frames) linked by a time-line. Keyframing involves taking a start frame and an end frame and generating the frames in between these two, in order to create the series of frames that can be perceived as an animation. This process, which we will look at later, is called 'tweening' or 'inbetweening'. Traditionally, in animation studios where animations are hand drawn, the head animator would draw the keyframes – which are the most important frames – and an assistant would draw the in between frames. It is possible to generate the series of frames digitally, or get the computer to do this, but it is important that you understand what is involved and how it is done.

4.4.4 Layering

Layering is an important technique that is used to save time in making animation. A background might remain static, and only the figures in it (or even only one of the figures) might need to move. In this case, it is more efficient to compose the animation of a number of layers, and only animate the layer or layers where movement needs to be seen.

Learning activity

Next time you watch an animation notice that the background is always more detailed than the characters. Can you see evidence of layering? How does layering work?

Thinking back to your work in *Processing*, and your experience of any other image manipulation packages, how does layering work in these?

4.4.5 Interpolation and curve-fitting

Figuring out what the in-between images need to look like can be tackled in a variety of ways. One approach that works is to use *interpolation*, and there are different types of interpolation that you will have seen in other contexts: for example, the demodulation performed on sampled audio data in Section 2.6.1 is a form of interpolation, as is the modelling step for digital audio compression in Section 2.6.2.

There are various kinds of interpolation for fitting curves, which include simple linear interpolation (where each point gets a straight line function to join it to the next) and cubic splines (where a polynomial function is fitted to the points). Bézier curves[3] do not actually fit a curve that goes through each point, but instead provide a function that comes close to each of the points in a principled and smooth fashion.

In the context of animation, consider two keyframes as specifying two positions of an object: the first keyframe specifies the start position, and the second the final position. Each point in the start keyframe will need to 'move' to its appropriate place in the end keyframe. Keyframes are settings for a value at a given time – a $\langle \text{time}, \text{value} \rangle$ tuple.

Linear interpolation between two positions $P(t_{k-1})$ and $P(t_k)$ is achieved by the

[3] Pierre Étienne Bézier (1910–1999), French engineer.

linear weighting of the two end positions; as time t moves between t_{k-1} and t_k, the interpolation parameter α moves linearly between 0 and 1, giving:

$$P(t) = \alpha P(t_k) + (1 - \alpha)P(t_{k-1}) \tag{4.1}$$

Writing this out explicitly in terms of the time variable t, we get

$$P(t) = \frac{t - t_{k-1}}{t_k - t_{k-1}} P(t_k) + \left(1 - \frac{t - t_{k-1}}{t_k - t_{k-1}}\right) P(t_{k-1}) \tag{4.2}$$

for $t_{k-1} < t < t_k$.

Learning activity

Use the above formula to create a simple animation between two keyframes, of a two-dimensional scene. Use *Processing* to realise the animation.

Think also about the timing of this animation, and make it run over two seconds, where you are showing 25 frames per second.

Linear interpolation will tend to produce a jerky animation, because (unless very many keyframes are used) the motion of the object being moved will change in a noticeably discontinuous fashion at the keyframe. In order to remove this effect, it is common to use an interpolation method where, at each keyframe, not only the position of the object is specified but also its rate of change (its velocity): in this way, the interpolation can produce a smooth motion.

One way of achieving this is to use *csplines* (or *cubic Hermite splines*[4]). Cubic Hermite splines provide a mechanism for interpolating, given an explicit specification of the velocity at each keyframe. However, animators do not typically specify the velocity themselves at keyframes; instead, a particular choice of velocity is computed from the positions at the preceding and subsequent keyframes:

$$m(t_k) = \frac{P(t_{k+1}) - P(t_{k-1})}{2} \tag{4.3}$$

yielding a particular spline interpolation called a Catmull-Rom spline.[5] This computed velocity (or *gradient*) is, by construction, in the direction defined by the the previous and next points, and so will naturally yield a smooth interpolation.

4.5 Visualisation

Visualisation is an important device in explaining and understanding concepts, and making sense of information. To visualise something involves making a mental image of it. Significant here is the term 'image' – visualisation works in the domain of the visual, and so some kind of picture or diagram will usually be a component.

Examples of visualisation include:

[4] Charles Hermite (1822–1901), French mathematician.

[5] Edwin Catmull (1945–), American computer scientist and president of Walt Disney Animation Studios and Pixar Animation Studios.

- The turning of statistical data into a distribution map. This has been used to great effect in illustrating important concepts, such as economic distribution globally; or population density across the world. It can be linked to other visual techniques, and can also help in research. A particular use is the combination of distribution maps, to identify commonalities in geographical areas, and this can be the start of research into causes and connections.
- Visual presentation of audio information. In its simplest form, an example is the display of a graphic equaliser (see *Creative computing II*, Vol. 1, Section 5.1.1), that can then be used to analyse and manipulate music.
- Presenting medical information in a visual form, such as in models of DNA.
- Using mental images to explain mathematical concepts, such as recursion.
- Building scale models of architectural designs, as well as creating digital visualisations of these.

Learning activity

Design a visualisation of a mathematical or arithmetic concept, that could be used as an aid to teaching that concept to students.

An example of visualisation, which itself includes some information about and examples of different kinds of visualisation methods, can be found at `http://www.visual-literacy.org/periodic_table/periodic_table.html`.

Visualisation can be combined with animation to create particular impacts. John Maeda makes use of this in visually creative design. See in particular his work for Shishedo clocks.[6]

Learning activity

Choose some statistical information about something in the country or city or town in which you live. Try to choose information in an area that is of particular interest to you.

Create a visualisation of this information. First create a static visualisation. Then think about whether it would be appropriate to incorporate any kind of animation into the visualisation, and if so, discuss what that would be. If possible and appropriate, implement it.

4.6 Summary and learning outcomes

This chapter provides a basic introduction to animation, and explains the connection between visual perception and animation. There is also some historical background on how animation has developed, and a discussion of animation in the digital world. Visualisation and its relation to animation is presented.

[6] Available at `http://www.maedastudio.com/index.php?category=kinetic`. Maeda's work is an excellent example of the synthesis of design aesthetic and informational impact: another example of this, though without animation, can be found at `http://www.maedastudio.com/2006/lifecounter/index.php?category=new&this=lifecounter&group=line`.

With a knowledge of the contents of this chapter and its directed reading and activities, you should be able to:

- describe the difference between stop motion animation and flat animation, and decide which is appropriate for different uses;
- describe how animation works, in terms of perception;
- describe different techniques involved in making animated films;
- discuss the role of visualisation in the conveying of information.

4.7 Exercises

1. Which type of animation is more 'life-like' – flat animation or stop-motion animation? Discuss both of these, and support your discussion with evidence from literature.
2. Find out about the work of these two animation studios: *Pixar* and *Claymation*. Give examples of work that each has produced.

   Aardman Animations is a British-based animation studio. Find out what you can about their work, and give an example of a film produced by them.

   Find some other animation studios and discuss their work.
3. The concept of 'persistence of vision' is often mentioned in descriptions of how animation is achieved. Find out what this is, and write a short essay describing the arguments around it. (You may wish to refer to Section 1.5.2.)
4. What is the difference between the 'frame rate' and the 'refresh rate' in the context of digital video? Are these concepts also relevant to celluloid film?
5. Imagine a digital animation that runs for 30 seconds. How many frames would it have? If the animation was a flat one, how many drawings would be required? Are there ways to reduce the number of drawings?
6. How would you create an animation using *Processing*? What is the difference between this, and the use of the *Processing* language statements to achieve 3-D movement?
7. Using *Processing*, create a simple animation that simulates a flip book.
8. What is the difference between animation and visualisation? What do they have in common? Discuss how animation and visualisation can be used in the field of architecture. Discuss how animation and visualisation can be used in the area of music.
9. What are Bézier curves, and how can they be used for tweening in animation? What are Hermite curves? Find out what the formulae for calculating these functions are, and try them out on simple keyframing examples.
10. What are 'slow-in' and 'slow-out', and how are they used to make animation less jerky?
11. In *Processing*, write a program that lets the user click on points on a screen. The time and position of each click are stored as an array of keyframes. When the user presses `Return`, the stored movements are replayed with a dot moving smoothly between the points.

    Perform the inbetweening using a variety of techniques (such as linear interpolation, Catmull-Rom splines, Bézier curves, etc.) and evaluate the results.

Appendix A

Sample examination paper

UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B. Sc. Examination (sample)

CREATIVE COMPUTING

2910227   Creative Computing 2

Duration: 3 hours

There are six questions in this paper; you should answer no more than FOUR questions. Full marks will be awarded for complete answers to a total of FOUR questions. Each question carries 25 marks; the marks for each part of a question are indicated at the end of the part in [.] brackets.

There are 100 marks available on this paper.

THIS EXAMINATION PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

Question 1 Colour blindness and colour spaces

(a) Explain the meaning of a 'confusion line' in the context of colour blindness and the CIE chromaticity diagram. [4]

(b) On the CIE chromaticity diagram, at what (x, y) co-ordinates do the lines joining confusable colours join up for:

 (i) protanopes; [2]

 (ii) deuteranopes. [2]

(c) What criterion on the colour space co-ordinates, in addition to the above, is there for a colour-blind observer to perceive no difference between two different colours? [2]

(d) Consider the following colours, expressed in CIE XYZ (standard observer) co-ordinates:

- c_1 = {68, 40, 292};
- c_2 = {75, 60, 165};
- c_3 = {64, 40, 96};
- c_4 = {64, 80, 56}.

Given that the intersection of the confusion lines for tritanopia in CIE chromaticity co-ordinates is $(0.18, 0.0)$:

 (i) transform the colour specifications into CIE chromaticity co-ordinates; [4]

 (ii) determine which, if any, pairs of colours from {c_1,...,c_4} lie on a tritanopic confusion line; [7]

 (iii) determine which of the colours from {c_1,...,c_4} are actually confusable by a tritanopic observer. [4]

Question 2 Multimedia information retrieval

(a) In the context of Multimedia information retrieval, what is a *feature*? [4]

(b) Give two examples of textual metadata features and two of content-based features suitable for use in a musical information retrieval system. [4]

(c) A collection of images is stored on disk; in addition, a linked list of $\langle \text{filename}, \text{colour} \rangle$ pairs is constructed, where the first element of the pair is the image's filename and the second is the average colour in CIE $L^*a^*b^*$ co-ordinates.

(i) Suggest an appropriate distance measure for comparing the colours in this format. [2]

(ii) Comment on the computational complexity for retrieving an item from this database, and the implications that this complexity has in practical application. [4]

(iii) The linked list contains three elements:

- i_1: $\langle \texttt{lena.jpg}, \{100, -2.4, -19\} \rangle$
- i_2: $\langle \texttt{ship.png}, \{53, -1.4, -12\} \rangle$
- i_3: $\langle \texttt{lute.tif}, \{56, 72, 43\} \rangle$.

Which of these files is the closest match to a query colour with co-ordinates $\{37, 43, 0\}$? [8]

(iv) In the light of the filenames, suggest which image would be most appropriate if in addition to specifying the query colour above, the user added 'boat' as a textual search term. Comment on your answer. [3]

Question 3 Signals and Digital audio files

(a) What is a signal? Describe the difference between discrete and continuous signals, giving two examples of each with different dimensionality. [8]

(b) Describe how a one-dimensional discrete-time signal can be represented as a vector in *Octave*. Include in your answer how to relate the *Octave* vector index to the corresponding moment in time. [4]

(c) The modelling step in lossless audio compression involves predicting the next sample given a certain number of previous samples.

  (i) Write down expressions for the constant (first-order) and linear (second-order) predictors. [3]

  (ii) Sketch graphs of the pulse-code modulation sampling procedure when the signal consists of:

  - a low-frequency sinusoid; [2]
  - a sinusoid with frequency close to the Nyquist frequency of the sampling process. [2]

  (iii) Hence or otherwise suggest which of the predictors will achieve better compression in the case of each of the two signals above. Explain your reasoning. [6]

Question 4 Hearing and sound

(a) Describe the mechanisms involved in hearing. Include a discussion on how the ear amplifies the effect of an incident pressure wave. [6]

(b) Intensity of sound is perceived approximately logarithmically.

(i) Explain what is meant by this statement. [3]

(ii) Describe how to measure the difference between sound pressures in decibels on a suitable scale. [3]

(iii) 2×10^{-5}Pa is the sound pressure for the auditory threshold at 2kHz; this can be the reference point for the loudness scale. Compute the loudness in decibels for a sound pressure of 3×10^{-3}Pa and for 6×10^{-1}Pa. [6]

(iv) Explain why 190dB is a limiting point for loudness on Earth. [2]

(c) Describe the sensitivity of human hearing to frequency. [5]

Question 5 Audio and image filtering

(a) Describe the phenomena of *echo* and *reverberation*, including in your answer the mechanisms by which they are naturally produced, and typical timescales for each of the phenomena. [8]

(b) 'Computing the effect of a Linear Time-Invariant system on a signal involves calculating the convolution of the system's impulse response with the signal.' In the context of systems, define the terms:

(i) linear; [3]

(ii) time-invariant; [3]

In your definitions include statements of the equivalences which hold in systems of the appropriate form.

(c) What issue arises in practice with an implementation of a system as a direct computation of the convolution, and how can this be addressed? [6]

(d) Briefly explain the effect of applying a high-pass filter to:

(i) a sound; [2]

(ii) an image. [3]

Question 6 Animation

(a) Describe the *flat* and *stop-motion* traditional animation techniques. [9]

(b) Relate the traditional animation techniques in part (a) to computer animation techniques. Include in your answer a discussion of the relationship between *keyframing* in traditional and digital animation. [6]

(c) Two successive keyframes in an animation involve an object having the following positions at the given times:

time/s	(x, y)
37	(30, 47)
45	(62, 39)

Assuming linear interpolation, compute the object's position at:

(i) 40s; [3]

(ii) 44s. [3]

(d) What problem does linear interpolation cause, and how is this normally addressed in digital animation? [4]

Index