Fundamentals of Multimedia
ISBN: 0130618721

Ze-Nian Li and Mark S. Drew
School of Computing Science
Simon Fraser University

Exercise Solutions

Contents

Chapter 1

Introduction to Multimedia

Exercises

1. Identify three novel applications of the Internet or multimedia applications. Discuss why you think these are novel.

   Answer:
   WML – Wireless markup Language.
   Mobile games: massive multiplayer online role-playing game (Mmorpg)
   Multisensory data capture
   Capture of context
   Represent and adjust recollections in memory over time
   "Fictive Art" in new media: beyond RPGs in the use of narrative and fictions to create made-up worlds, imaginary situations, and odd situations (an example is "The Museum of Jurassic Technology").
   Bridging the semantic gap problem in automatic content annotation systems — the gulf between the semantics that users expect and the low-level features (content descriptions) that systems actually use: one solution is "an approach called computational media aesthetics. We define this approach as the algorithmic study of a variety of image, space, and aural elements employed in media ... based on the media elements usage patterns in production and the associated computational analysis of the principles that have emerged while clarifying, intensifying, and interpreting some event for the audience." (IEEE Multimedia Magazine, Volume: 10, Issue: 2, Year: April-June 2003)

2. Briefly explain, in your own words, "Memex" and its role regarding hypertext. Could we carry out the Memex task today? How do you use Memex ideas in your own work?

   Answer:
   Memex was a theoretical system explicated by Vanevar Bush in a famous 1945 essay. His main ideas involved using *associative memory* as an aid for organizing a weler of material. He even adumbrated the concept of *links*.

3. Your task is to think about the transmission of smell over the Internet. Suppose we have a smell sensor at one location and wish to transmit the *Aroma Vector* (say) to a receiver to reproduce the same sensation. You are asked to design such a system. List three key issues to consider and two applications of such a delivery system. *Hint*: Think about medical applications.

Answer:
"October 6, 2000 – DigiScents, Inc., the pioneer of digital scent technology, received the 'Best New Technology' award for its iSmell(TM) device at the 'Best of RetailVision Awards' at the Walt Disney World Dolphin Hotel in Lake Buena Vista, Fla. Retailers such as BestBuy, RadioShack, CompUSA and other industry giants voted on the vendor awards."
"DigiScents ... The company would send you a dispenser about the size of a computer speaker. You'd plug it into your PC. It would be filled with chemicals that, when mixed, could recreate most any smell. Tiny bits of data would come in over the Net to tell your dispenser what smell to make. There would be a portal where you could find scents. DigiScents calls it – and at first I thought they were joking – a 'Snortal.'"

4. Tracking objects or people can be done by both sight and sound. While vision systems are precise, they are relatively expensive; on the other hand, a pair of microphones can detect a person's *bearing* inaccurately but cheaply. Sensor *fusion* of sound and vision is thus useful. Surf the web to find out who is developing tools for video conferencing using this kind of multimedia idea.

   Answer:
   "Distributed Meetings: A Meeting Capture and Broadcasting System," Ross Cutler, Yong Rui, Anoop Gupta, JJ Cadiz Ivan Tashev, Li-wei He, Alex Colburn, Zhengyou Zhang, Zicheng Liu, Steve Silverberg, Microsoft Research, ACM Multimedia 2002,
   http://research.microsoft.com/research/coet/V-Kitchen/chi2001/paper.pdf

5. *Non-photorealistic* graphics means computer graphics that do well enough without attempting to make images that look like camera images. An example is conferencing (let's look at this cutting-edge application again). For example, if we track lip movements, we can generate the right animation to fit our face. If we don't much like our own face, we can substitute another one — facial-feature modeling can map correct lip movements onto another model. See if you can find out who is carrying out research on generating avatars to represent conference participants' bodies.

   Answer:
   See: anthropic.co.uk

6. Watermarking is a means of embedding a hidden message in data. This could have important legal implications: Is this image copied? Is this image doctored? Who took it? Where? Think of "messages" that could be sensed while capturing an image and secretly embedded in the image, so as to answer these questions. (A similar question derives from the use of cell phones. What could we use to determine who is putting this phone to use, and where, and when? This could eliminate the need for passwords.)

   Answer:
   Embed retinal scan plus date/time, plus GPS data; sense fingerprint.

Chapter 2

Multimedia Authoring and Tools

Exercises

1. What extra information is multimedia good at conveying?

   (a) What can spoken text convey that written text cannot?

   Answer:
   Speed, rhythm, pitch, pauses, etc...
   Emotion, feeling, attitude ...

   (b) When might written text be better than spoken text?

   Answer:
   Random access, user-controlled pace of access (i.e. reading vs. listening)
   Visual aspects of presentation (headings, indents, fonts, etc. can convey information)
   For example: the following two pieces of text may sound the same when spoken:
   I said "quickly, come here."
   I said quickly "come here."

2. Find and learn 3D Studio Max in your local lab software. Read the online tutorials to see this software's approach to a 3D modeling technique. Learn texture mapping and animation using this product. Make a 3D model after carrying out these steps.

3. Design an interactive web page using Dreamweaver. HTML 4 provides layer functionality, as in Adobe Photoshop. Each layer represents an HTML object, such as text, an image, or a simple HTML page. In Dreamweaver, each layer has a marker associated with it. Therefore, highlighting the layer marker selects the entire layer, to which you can apply any desired effect. As in Flash, you can add buttons and behaviors for navigation and control. You can create animations using the Timeline behavior.

4. In regard to automatic authoring,

   (a) What would you suppose is meant by the term "active images"?

   Answer:
   Simple approach: Parts of the image are clickable.
   More complex: Parts of the image have knowledge about themselves.

   (b) What are the problems associated with moving text-based techniques to the realm of image-based automatic authoring?

Answer:
Automatic layout is well established, as is capture of low-level structures such as images and video. However amalgamating these into higher-level representations is not well understood, nor is automatically forming and linking appropriate level anchors and links.

(c) What is the single most important problem associated with automatic authoring using legacy (already written) text documents?

Answer:
Overwhelming number of nodes created, and how to manage and maintain these.

5. Suppose we wish to create a simple animation, as in Fig. 2.30. Note that this image is exactly

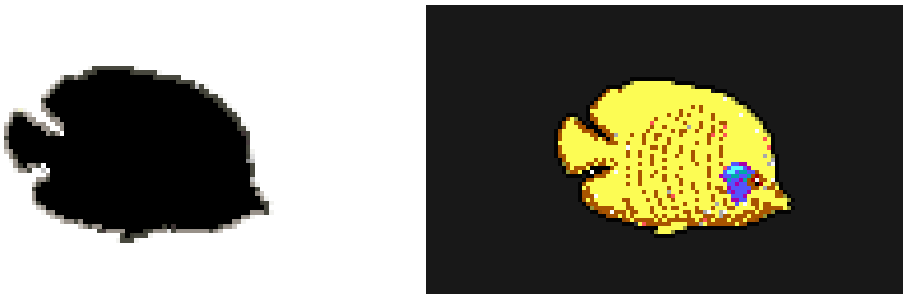
Fig. 2.30: Sprite, progressively taking up more space.

what the animation looks like at some time, not a figurative representation of the *process* of moving the fish; the fish is repeated as it moves. State what we need to carry out this objective, and give a simple pseudocode solution for the problem. Assume we already have a list of (x, y) coordinates for the fish path, that we have available a procedure for centering images on path positions, and that the movement takes place on top of a video.

Answer:

```
\\ We have a fish mask as in Figure \ref{FIG:MASKANDSPRITE}(a), and
\\ also a fish sprite as in Figure \ref{FIG:MASKANDSPRITE}(b).
\\ Fish positions have centers posn(t).x posn(t).y

currentmask = an all-white image
currentsprite = an all-black image
for t = 1 to maxtime {
   \\ Make a mask fishmask with the fish mask black area
   \\ centered on position posn(t).x, posn(t).y
   \\ and a sprite fishsprite with the colored area also moved
   \\ to posn(t).x, posn(t).y
   \\ Then expand the mask:
   currentmask = currentmask AND fishmask \\ enlarges mask
   currentsprite = currentsprite OR fishsprite \\ enlarges sprite
   \\ display current frame of video with fish path on top:
```

```
    currentframe = (frame(t) AND currentmask) OR currentsprite
}
```



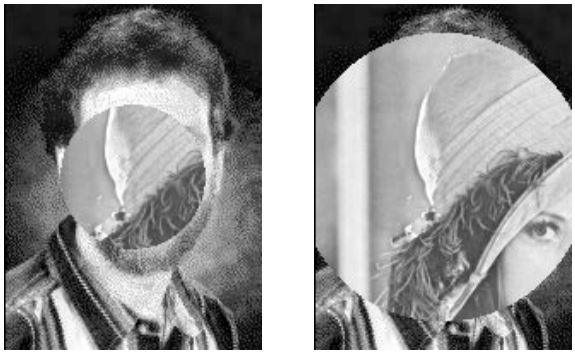
(a) (b)

Fig. 2.30: (answer) Mask and Sprite.

6. For the slide transition in Fig. 2.11, explain how we arrive at the formula for x in the unmoving right video R_R.

   Answer:
   if $x/x_{max} \geq t/t_{max}$, then we are in the right-hand video. The value of x is to the right of x_T, and the value in the *unmoving* right-hand video is that value of x, reduced by x_T so that we are in units with respect to the left of the right-hand video frame. That is, in the right-hand video frame we are at position $x - x_t$, which is $x - (x_{max} * t/t_{max})$.

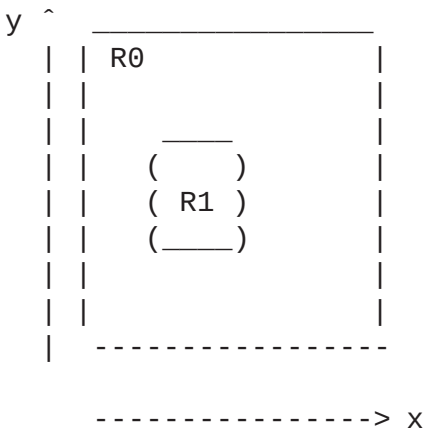
7. Suppose we wish to create a video transition such that the second video appears under the first video through an opening circle (like a camera iris opening), as in Figure 2.31. Write a formula to use the correct pixels from the two videos to achieve this special effect. Just write your answer for the red channel.



(a) (b)

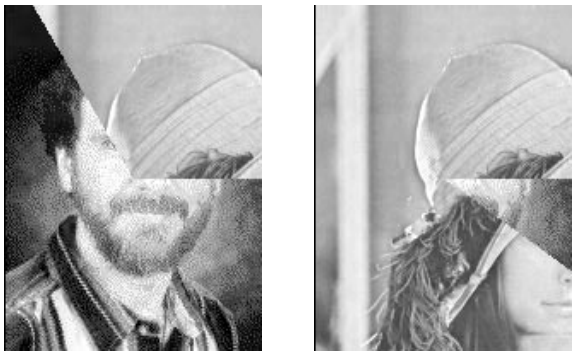
Fig. 2.31: Iris wipe: (a): Iris is opening. (b): At a later moment.

Answer:




```
radius of transition r_T = 0 at time t = 0

r_T = r_max = sqrt( (x_max/2)^ + (y_max/2)^2 )  at time t=t_max

-->  r_T = r_max * t / t_max

At x,y,
  r = sqrt(  (x-x_max/2)^2 + (y-y_max/2)^2  )

  If (  r < (t/t_max)*r_max  )
     R(x,y,t) = R1(x,y,t)
  Else
     R(x,y,t) = R0(x,y,t)
```

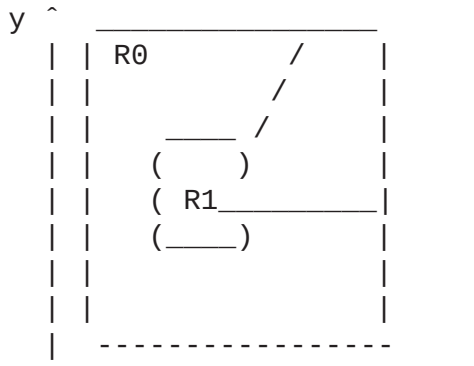
8. Now suppose we wish to create a video transition such that the second video appears under the first video through a moving radius (like a clock hand), as in Figure 2.32. Write a formula to use the correct pixels from the two videos to achieve this special effect for the red channel.



(a) (b)

Fig. 2.32: Clock wipe: (a): Clock hand is sweeping out. (b): At a later moment.

Answer:




```
angle of transition a_T = 0 at time t = 0

a_T = a_max = 360  at time t=t_max

-->  a_T = a_max * t / t_max


At x,y,
  a = atan(  -(y-y_max/2)/(x-x_max/2)  )
  \\ Since y in defn of angle increases from bottom, not from top
  \\ like rows. Watch for correct quadrant, though--use 2-arg atan.

  If (  a < (t/t_max)*a_max  )
     R(x,y,t) = R1(x,y,t)
  Else
     R(x,y,t) = R0(x,y,t)
```

9. Suppose you wish to create a wavy effect, as in Figure 2.33. This effect comes from replacing the image x value by an x value offset by a small amount. Suppose the image size is 160 rows $\times 120$ columns of pixels.

   (a) Using float arithmetic, add a sine component to the x value of the pixel such that the pixel takes on an RGB value equal to that of a different pixel in the original image. Make the maximum shift in x equal to 16 pixels.

   Answer:
   R = R(x + sin(y/120) * 16 , y) and similarly for G, B.

   (b) In Premiere and other packages, only integer arithmetic is provided. Functions such as `sin` are redefined so as to take an `int` argument and return an `int`. The argument to the `sin` function must be in $0\,..\,1{,}024$, and the value of `sin` is in $-512\,..\,512$: `sin(0)` returns `0`, `sin(256)` returns `512`, `sin(512)` returns `0`, `sin(768)` returns `-512` and `sin(1,024)` returns `0`. Rewrite your expression in part (a) using integer arithmetic.

Fig. 2.33: Filter applied to video.

Answer:

```
R = R(x + sin( (y*1024)/120 ) /32,y) and similarly
for G,B.
[In Premiere:     src(x + sin( (y*1024)/120 ) /32,y ,p)    ]
Why: y in 0..119;  (y*1024)/120 in 0..1023; the resulting  sin
is in -512..512;  and dividing by 32 puts the offset range into
-16..16.
```

(c) How could you change your answer to make the waving time-dependent?

Answer:

```
R = R(x +  t*sin(y*(1024/120) ) /(32*tmax),y)
[In Premiere: src(x +  t*sin(y*(1024/120) ) /(32*tmax),y ,p) ]
Note the order: else have t/tmax==0 or 1 only.
```

10. How would you create the image in Figure 2.6? Write a small program to make such an image. *Hint:* Place R, G, and B at the corners of an equilateral triangle inside the circle. It's best to go over all columns and rows in the output image rather than simply going around the disk and trying to map results back to (x, y) pixel positions.

Answer:

```
% Matlab script:
SIZE = 256;
im = ones(SIZE,SIZE,3);
% Place R at (0,1).
```

```
% Place G at 120 degrees.
% Place B at 240 degrees.
% The outside perimeter goes from R to G as we go from
%   R to G.
% And from B to R as we go from 240 to 360.
%
% At a position where the Outside Perimeter value
% is   out  , at radius   r  the color is
% (1-r)*(1,1,1) + r*(out)

% Go over all pixels:
for j = 1:SIZE
for i = 1:SIZE
  x = j-SIZE/2;
  y = i-SIZE/2;
  r = sqrt(x*x+y*y);
  if (r<=(SIZE/2))
    ang = 180/pi*atan2(y,x);
    if ang <0
      ang = 360+ang;
    end
    if ang < 120 % between R and G
      out = [(120-ang)/120 ; ang/120 ; 0];
    elseif ang < 240 % between G and B
      out = [0 ; (240-ang)/120 ; (ang-120)/120];
    else % between B and R
      out = [(ang-240)/120 ; 0 ; (360-ang)/120];
    end; % if ang
    % and could make the in-between bands broader by not using
    % linear interpolation, if wished.

    %linear:
      im(i,j,:) = ((SIZE/2)-r)/(SIZE/2)*[1;1;1] + r/(SIZE/2)*out;
    % and normalize the color to bright:
    temp = max( im(i,j,:) );
    im(i,j,:) = im(i,j,:)/temp; % takes one channel to 1.0
  end; % if r

end
end
imshow(im)

imwrite(im,'colorwheel256.bmp');
```

11. As a longer exercise for learning existing software for manipulating images, video, and music, make a 1-minute digital video. By the end of this exercise, you should be familiar with PC-based equipment and know how to use Adobe Premiere, Photoshop, Cakewalk Pro Audio, and other multimedia software.

(a) Capture (or find) at least three video files. You can use a camcorder or VCR to make your own (through Premiere or the like) or find some on the Net.

(b) Compose (or edit) a small MIDI file with Cakewalk Pro Audio.

(c) Create (or find) at least one WAV file. You may either digitize your own or download some from the net.

(d) Use Photoshop to create a title and an ending.

(e) Combine all of the above to produce a movie about 60 seconds long, including a title, some credits, some soundtracks, and at least three transitions. Experiment with different compression methods; you are encouraged to use MPEG for your final product.

(f) The above constitutes a minimum statement of the exercise. You may be tempted to get very creative, and that's fine, but don't go overboard and take too much time away from the rest of your life!

Chapter 3

Graphics and Image Data Representations

Exercises

1. Briefly explain why we need to be able to have less than 24-bit color and why this makes for a problem. Generally, what do we need to do to adaptively transform 24-bit color values to 8-bit ones?

   Answer:

   May not be able to handle such large file sizes or not have 24-bit displays.
   The colors will be somewhat wrong, however.
   We need to cluster color pixels so as to best use the bits available to be as accurate as possible for the colors in an image. In more detail: variance minimization quantization–vmquant.m Minimum variance quantization allocates more of the available colormap entries to colors that appear frequently in the input image and allocates fewer entries to colors that appear infrequently. Therefore if there are for example many reds, as in a red apple, there will be more resolution in the red part of the color cube. An excellent implementation of this idea is Wu's Color Quantizer (see Graphics Gems vol. II, pp. 126-133).

2. Suppose we decide to quantize an 8-bit grayscale image down to just 2 bits of accuracy. What is the simplest way to do so? What ranges of byte values in the original image are mapped to what quantized values?

   Answer:
   Just use the first 2 bits in the grayscale value.

   ```
   I.e., any values in the ranges
   0000 0000 to 0011 1111
   0100 0000 to 0111 1111
   1000 0000 to 1011 1111
   1100 0000 to 1111 1111
   are mapped into 4 representative grayscale values.
   In decimal, these ranges are:
   0 to (2^6-1)
   2^6 to (2^7-1)
   2^7 to 2^7 + (2^6-1)
   2^7 + 2^6 to (2^8-1)
   i.e.,
   ```

```
0 to 63
64 to 127
128 to 191
192 to 255
Then reconstruction values should be taken as the middle
of these ranges; i.e.,
32
96
160
224
```

3. Suppose we have a 5-bit grayscale image. What size of ordered dither matrix do we need to display the image on a 1-bit printer?

   Answer:

   ```
   2^5=32 levels ~= n^2+1 with n=6; therefore need D(6)
   ```

4. Suppose we have available 24 bits per pixel for a color image. However, we notice that humans are more sensitive to R and G than to B — in fact, 1.5 times more sensitive to R or G than to B. How could we best make use of the bits available?

   Answer:
   ratio is 3:3:2, so use bits 9:9:6 for R:G:B.

5. At your job, you have decided to impress the boss by using up more disk space for the company's grayscale images. Instead of using 8 bits per pixel, you'd like to use 48 bits per pixel in RGB. How could you store the original grayscale images so that in the new format they would appear the same as they used to, visually?

   Answer:
   48 bits RGB means 16 bits per channel: so re-store the old ints, which were $< 2^8$, as new ints $< 2^{16}$. But then the new values have to be created by multiplying the old values by 2^8, so that e.g. a mid-gray is still a mid-gray. As well, have to duplicate the old gray into all three of R,G,B.

6. Sometimes bitplanes of an image are characterized using an analogy from mapmaking called "elevations". Figure 3.18 shows some elevations.

   Suppose we describe an 8-bit image using 8 bitplanes. Briefly discuss how you could view each bitplane in terms of geographical concepts.

   Answer:
   We can think of the 8-bit image as a set of 1-bit *bit-planes*, where each plane consists of a 1-bit representation of the image at higher and higher levels of 'elevation': a bit is turned on if the image pixel has a value that is at or above that bit level.

7. For the color LUT problem, try out the median-cut algorithm on a sample image. Explain briefly why it is that this algorithm, carried out on an image of red apples, puts more color gradation in the resulting 24-bit color image where it is needed, among the reds.

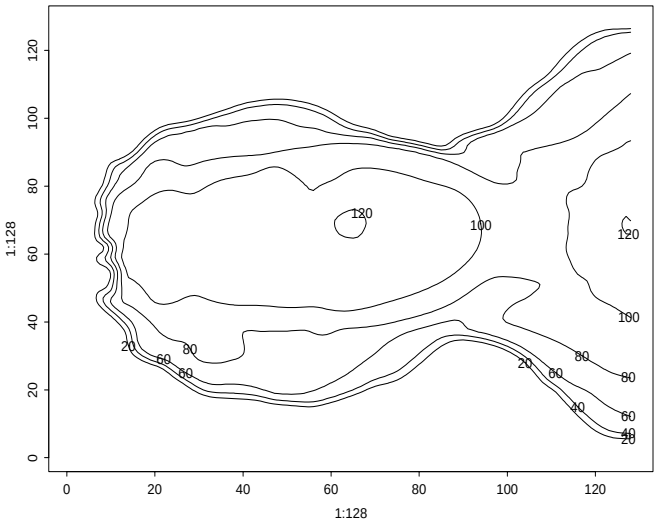


Fig. 3.18: Elevations in geography.

8. In regard to nonordered dithering, a standard graphics text [2] states, "Even larger patterns can be used, but the spatial versus intensity resolution trade-off is limited by our visual acuity (about one minute of arc in normal lighting)."

   (a) What does this sentence mean?

   Answer:
   If we increase the matrix size to $n \times n$, with a larger n, then we make the number of intensity levels available, $n^2 + 1$, greater and thus increase the intensity resolution. But then the size of the patterns laid down in bi-level dots gets larger, decreasing the spatial resolution — less detail is in each small area. The larger is the matrix of patterns, the greater is the possibility that we can see gaps between dots. We would prefer output that did not let us see the gaps between bi-level dots: this is fixed by our ability to detect the space between small dots that are separated by one minute of arc.

   (b) If we hold a piece of paper out at a distance of 1 foot, what is the approximate linear distance between dots? (*Information*: One minute of arc is 1/60 of one degree of angle. Arc length on a circle equals angle (in radians) times radius.) Could we see the gap between dots on a 300 dpi printer?

   Answer:
   One minute of arc is $1/60 * \pi/180$ radians, and at $r = 25$ cm, the arc length is $25 * \pi/(60 * 180)$, or about $25 * 3/10800 \sim = 1/100$ cm. For a 300 dpi printer, the dot separation is about 100 d.p.cm, so we could just see such a gap (this is actually affected by what surrounds the image).

9. Write down an algorithm (pseudocode) for calculating a color histogram for RGB data.

Answer:

```
int hist[256][256][256];
image is an appropriate struct with int fields red,green,blue

for i=0..(MAX_Y-1)
  for j=0..(MAX_X-1)

     R = image[x][y].red;
     G = image[x][y].green;
     B = image[x][y].blue;
     hist[R][G][B]++;
```

Chapter 4

Color in Image and Video

Exercises

1. Consider the following set of color-related terms:

   (a) wavelength

   (b) color level

   (c) brightness

   (d) whiteness

   How would you match each of the following (more vaguely stated) characteristics to each of the above terms?

   (a) luminance $\Rightarrow$ **brightness**

   (b) hue $\Rightarrow$ **wavelength**

   (c) saturation $\Rightarrow$ **whiteness**

   (d) chrominance $\Rightarrow$ **color level**

2. What color is outdoor light? For example, around what wavelength would you guess the peak power is for a red sunset? For blue sky light?

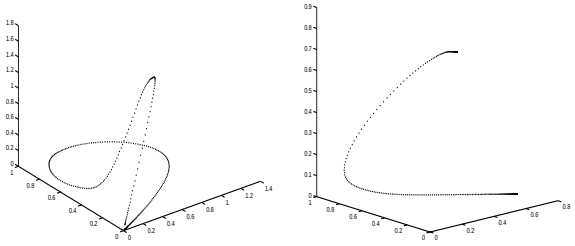
   Answer:
   450 nm, 650 nm.

3. "The LAB gamut covers all colors in the visible spectrum."

   (a) What does this statement mean? Briefly, how does LAB relate to color? Just be descriptive.

   (b) What are (roughly) the relative sizes of the LAB gamut, the CMYK gamut, and a monitor gamut?

   Answer:
   CIELAB is simply a (nonlinear) restating of XYZ tristimulus values. The objective of CIELAB is to develop a more perceptually uniform set of values, for which equal distances in different parts of gamut imply roughly equal differences in perceived color. Since XYZ encapsulates a statement about what colors can in fact be seen by a human observer, CIELAB also "covers all colors in the visible spectrum."



(a) (b)

Fig. 4.20: (a): Color matching functions; (b): Transformed color matching functions.

XYZ, or equivalently CIELAB, by definition covers the whole human visual system gamut. In comparison, a monitor gamut covers just the triangle joining the R, G, and B pure-phosphor-color corners, so is much smaller. Usually, a printer gamut is smaller again, although some parts of it may overlap the boundary of the monitor gamut and thus allow printing of colors that in fact cannot be produced on a monitor. Printers with more inks have larger gamuts. (Incidentally, color slide films have considerably larger gamuts.)

4. Where does the chromaticity "horseshoe" shape in Figure 4.11 come from? Can we calculate it? Write a small pseudocode solution for the problem of finding this so-called "spectrum locus". *Hint*: Figure 4.20(a) shows the color-matching functions in Figure 4.10 drawn as a set of points in three-space. Figure 4.20(b) shows these points mapped into another 3D set of points. *Another hint*: Try a programming solution for this problem, to help you answer it more explicitly.

   Answer:
   The $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$, color-matching curves define the human visual system response to spectra. The outer boundary corresponds to the "spectrum locus", i.e., the response to a pure (laser-like) single wavelength sample. Interior points on the surface correspond to mixtures of wavelengths. To make visualization simpler, we go to a 2D chromaticity color space: $\{x, y\} = \{X, Y\}/(X + Y + Z)$. The boundary of the XYZ plot, projected to 2D, resembles a horseshoe shape.

```
% matlab script
load 'figs/chap4/xyz.dat' -ascii % 301 by 3 values
waves = (400:700)';
plot(waves, xyz)
plot3(xyz(:,1), xyz(:,2), xyz(:,3), '.');
%
denom = xyz(:,1) + xyz(:,2) + xyz(:,3);
xy = zeros(301,3); % declare
for k=1:3
  xy(:,k) = xyz(:,k) ./ denom;
end
plot3(xy(:,1), xy(:,2), xy(:,3), '.');
```

5. Suppose we use a new set of color-matching functions $\bar{x}^{new}(\lambda)$, $\bar{y}^{new}(\lambda)$, $\bar{z}^{new}(\lambda)$ with values

λ (nm)	$\bar{x}^{new}(\lambda)$	$\bar{y}^{new}(\lambda)$	$\bar{z}^{new}(\lambda)$
450	0.2	0.1	0.5
500	0.1	0.4	0.3
600	0.1	0.4	0.2
700	0.6	0.1	0.0

In this system, what are the chromaticity values (x, y) of equi-energy white light $E(\lambda)$ where $E(\lambda) \equiv 1$ for all wavelengths λ? Explain.

Answer:
The chromaticity values (x, y) are made from the XYZ triple $X = \sum_\lambda[\bar{x}(\lambda) \star E(\lambda)]$, $Y = \sum_\lambda[\bar{y}(\lambda) \star E(\lambda)]$, $Z = \sum_\lambda[\bar{z}(\lambda) \star E(\lambda)]$. For the new color-matching functions, since every $E(\lambda)$ is 1 for equi-energy white light, we form X, Y, Z via $\sum(\bar{x}), \sum(\bar{y}), \sum(\bar{z})$ = $(1, 1, 1)$, according to the values in the table; so the chromaticity is $x = X/(X + Y + Z)$ = $1/3$, and also $y = 1/3$.

6. (a) Suppose images are *not* gamma corrected by a camcorder. Generally, how would they appear on a screen?

   Answer:
   Too dark at the low-intensity end.

   (b) What happens if we artificially increase the output gamma for stored image pixels? (We can do this in Photoshop.) What is the effect on the image?

   Answer:
   Increase the number of bright pixels — we increase the number of pixels that map to the upper half of the output range. This creates a lighter image. – and incidentally, we also decrease highlight contrast and increase contrast in the shadows.

7. Suppose image file values are in $0\,..\,255$ in each color channel. If we define $\overline{R} = R/255$ for the red channel, we wish to carry out gamma correction by passing a new value $\overline{R}'$ to the display device, with $\overline{R}' \simeq \overline{R}^{1/2.0}$.

   It is common to carry out this operation using integer math. Suppose we approximate the calculation as creating new integer values in $0\,..\,255$ via

   $$(int)\,(255 \cdot\ (\overline{R}^{1/2.0}))$$

   (a) Comment (very roughly) on the effect of this operation on the number of actually available levels for display. *Hint*: Coding this up in any language will help you understand the mechanism at work better — and will allow you to simply count the output levels.

   (b) Which end of the levels $0\,..\,255$ is affected most by gamma correction — the low end (near 0) or the high end (near 255)? Why? How much at each end?

   Answer:

   (a) The integer values actually taken on are not as many as 256. (the number of levels comes out to 193). The reason is that some of the gamma corrected integer values equal the same integer value after truncation.

   (b) At the low end, the integer value $R = 0$ corresponds to the quantized value 0, whereas the int value 1 corresponds to

```
(int) ( 255* sqrt( 1/255 ) ) ~= (int) ( 255/16 ) = 15
```

so that an image value of 1 becomes the gamma-corrected value 15.

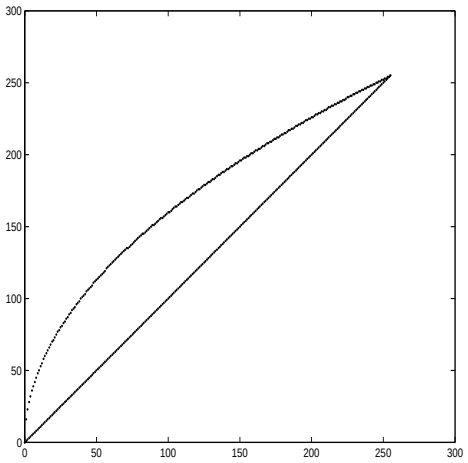
At the high end, $255 \mapsto 255$ **correctly, but** $254 \mapsto 255 * (1 - 1/255) \wedge 0.5 \simeq 255 * (1 - 0.5 * 1/255) = 255 - 0.5 \simeq 254$**, so that the high end is much less affected.**

Therefore overall the gamma-quantization greatly reduces the resolution of quantization at the low end of image intensity. The first few levels are 15, 22, 27, 31 ... **and the final few are 253, 254, 255.**

```
% matlab script:
% gamma.m
levels = 0:255;
levels = levels/255.0;
levels = floor(255* ( levels.^0.5 ) );
plot(0:255,levels)
levels = unique(levels);
 length(levels) % 193
 levels(1:5) % 0    15    22    27    31
```

8. In many computer graphics applications, γ-correction is performed only in color LUT (lookup table). Show the first five entries of a color LUT meant for use in γ-correction. *Hint*: Coding this up saves you the trouble of using a calculator.

   Answer:
   Use `round` **instead, but the idea is as in last question, applied now to a LUT.** $V' = V^{\frac{1}{\gamma}}$
   For answers: Do a LUT table for R G B, 256 rows, say.
   First row, indexed by 0, all entries 0. Any other kth row,
   $R = G = B = round(255 * (k/255)^{\frac{1}{\gamma}})$
   First five: 0 16 23 28 32



9. Devise a program to produce Figure 4.21, showing the color gamut of a monitor that adheres to SMPTE specifications.

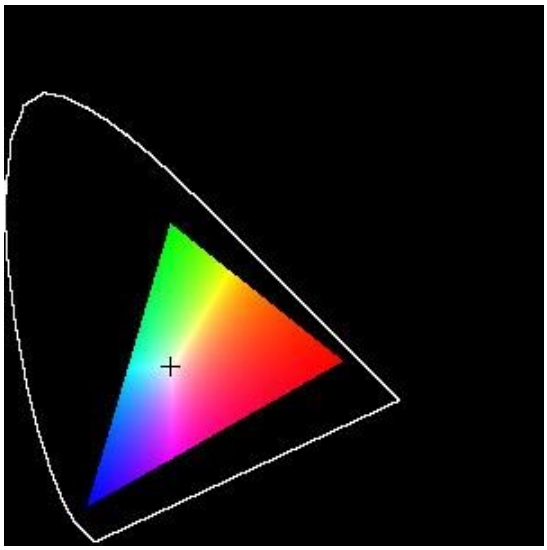


Fig. 4.21: SMPTE Monitor Gamut

Answer:
The simplest answer is derived from a Fortran program on the net:
http://www.physics.sfasu.edu/astro/color/chromaticity.html
Alterations are written in lower case. This code produces a `.ppm` **(portable pixmap) file in ASCII, which can be read by** `xv` **under Unix, for example.**
For C code, see after the Fortran code, on page 24.

```
c       FORTRAN:
c       Reconstructionist version of chromaticity diagram program
C        by Dan Bruton (astro@tamu.edu)
c        http://www.physics.sfasu.edu/astro/color/chromaticity.html
c
c       This program will create a ppm (portable pixmap)
c       image of an approximate chromaticity diagram using
c       equations from the Color Equations FAQ at
c       ftp://ftp.wmin.ac.uk/pub/itrg/coloureq.txt
c       or the Color Space FAQ.
c
        IMPLICIT REAL*8 (a-h,o-z)

c       Width, height and color depth of the ppm image, gamma
        PARAMETER(M=300)
        PARAMETER(N=M)
        PARAMETER(L=255)

        PARAMETER(GAM=1.0)

        DIMENSION CV(M,M,3), WXY(2,82)
        DIMENSION iCV(M,M,3)
```

```
c
c      Chromaticity Coodinates (x and y) for wavelengths in 5 nm
c      increments from 380 nm to 780 nm.
c
    DATA ((WXY(I,J),I=1,2),J=1,81)/
& 0.1741,0.0050, 0.1740,0.0050, 0.1738,0.0049, 0.1736,0.0049,
& 0.1733,0.0048, 0.1730,0.0048, 0.1726,0.0048, 0.1721,0.0048,
& 0.1714,0.0051, 0.1703,0.0058, 0.1689,0.0069, 0.1669,0.0086,
& 0.1644,0.0109, 0.1611,0.0138, 0.1566,0.0177, 0.1510,0.0227,
& 0.1440,0.0297, 0.1355,0.0399, 0.1241,0.0578, 0.1096,0.0868,
& 0.0913,0.1327, 0.0687,0.2007, 0.0454,0.2950, 0.0235,0.4127,
& 0.0082,0.5384, 0.0039,0.6548, 0.0139,0.7502, 0.0389,0.8120,
& 0.0743,0.8338, 0.1142,0.8262, 0.1547,0.8059, 0.1929,0.7816,
& 0.2296,0.7543, 0.2658,0.7243, 0.3016,0.6923, 0.3373,0.6589,
& 0.3731,0.6245, 0.4087,0.5896, 0.4441,0.5547, 0.4788,0.5202,
& 0.5125,0.4866, 0.5448,0.4544, 0.5752,0.4242, 0.6029,0.3965,
& 0.6270,0.3725, 0.6482,0.3514, 0.6658,0.3340, 0.6801,0.3197,
& 0.6915,0.3083, 0.7006,0.2993, 0.7079,0.2920, 0.7140,0.2859,
& 0.7190,0.2809, 0.7230,0.2770, 0.7260,0.2740, 0.7283,0.2717,
& 0.7300,0.2700, 0.7311,0.2689, 0.7320,0.2680, 0.7327,0.2673,
& 0.7334,0.2666, 0.7340,0.2660, 0.7344,0.2656, 0.7346,0.2654,
& 0.7347,0.2653, 0.7347,0.2653, 0.7347,0.2653, 0.7347,0.2653,
& 0.7347,0.2653, 0.7347,0.2653, 0.7347,0.2653, 0.7347,0.2653,
& 0.7347,0.2653, 0.7347,0.2653, 0.7347,0.2653, 0.7347,0.2653,
& 0.7347,0.2653, 0.7347,0.2653, 0.7347,0.2653, 0.7347,0.2653,
& 0.7347,0.2653/

       WXY(1,82)=WXY(1,1)
       WXY(2,82)=WXY(2,1)
c
c      Chromaticity Coordinates for Red, Green, Blue phosphors
c      and White Point=D65
C      SMPTE-C RGB:
c
       XR=0.63
       YR=0.34
       XG=0.31
       YG=0.595
       XB=0.155
       YB=0.07
       xw = 0.312713
       yw = 0.329016
       ZR=1.D0-(XR+YR)
       ZG=1.D0-(XG+YG)
       ZB=1.D0-(XB+YB)
       ZW=1.D0-(XW+YW)
```

```
c
c      Draw horseshoe outline
c
       DO II=1,M
        DO JJ=1,N
         DO KK=1,3
           CV(II,JJ,KK)=0
           iCV(II,JJ,KK)=0
         ENDDO
        ENDDO
       ENDDO

       DO J=1,81
         S1=REAL(M)*WXY(1,J)
         S2=REAL(M)*WXY(1,J+1)
         T1=REAL(N)*(1.D0-WXY(2,J))
         T2=REAL(N)*(1.D0-WXY(2,J+1))
     if ((s2-s1).ne.0.0) then
            SLOPE=(T2-T1)/(S2-S1)
     else
        s2 = s2 + 0.00001
            SLOPE=(T2-T1)/(S2-S1)
     endif
         I1=INT(S1)
         I2=INT(S2)
         DO II=I1,I2,JISIGN(1,I2-I1)
          S=REAL(II)
              J1=J2
          J2=INT(T1+SLOPE*(S-S1))
          IF ((J1.NE.0).AND.(J2.NE.0)) THEN
           DO JJ=J1,J2,JISIGN(1,J2-J1)
            DO KK=1,3
             iCV(II,JJ,KK)=1
            ENDDO
           ENDDO
          ENDIF
         ENDDO
       ENDDO
```

```
c
c      Calculate RGB Values for x and y coordinates
c
       rmax = 0.0
       do J=1,N
        do I=1,M
            X=REAL(1.*I/M)
            Y=REAL(1.*(N-J)/N)+0.00001
        YY=1.0
            XX=X*YY/Y
            ZZ=(1.-X-Y)*YY/Y
c smpte-c
            R=(3.5058*XX)-(1.7397*YY)-(0.5440*ZZ)
            G=-(1.0690*XX)+(1.9778*YY)+(0.0352*ZZ)
            B=(0.0563*XX)-(0.1970*YY)+(1.0501*ZZ)
C   print*, j,i,xx,yy,zz,r,g,b
        if (.not.( (iCV(I,J,1).eq.1).and.(iCV(I,J,1).eq.1).and.
           (iCV(I,J,1).eq.1))) then
           if ((R.LT.0.).OR.(G.LT.0.).OR.(B.LT.0.)) then
                  R=0.
                  G=0.
                  B=0.
           else
                  R=R**GAM
                  G=G**GAM
                  B=B**GAM
C                 Have same chromaticity if rescale RGB, so brighten:
                  rmax = max(r,g,b)
          r = r/rmax
          g = g/rmax
          b = b/rmax
           endif
               CV(I,J,1)=R
               CV(I,J,2)=G
               CV(I,J,3)=B
        endif
         enddo
       enddo

C      white point: draw a plus sign
       i = int(xw*M)
       j = int(N - (yw-0.00001)*N)
       do ii = i-5,i+5
          CV(ii,j,1) = 0D0
          CV(ii,j,2) = 0D0
          CV(ii,j,3) = 0D0
       enddo
       do jj = j-5,j+5
          CV(i,jj,1) = 0D0
          CV(i,jj,2) = 0D0
          CV(i,jj,3) = 0D0
       enddo
```

```
C       print*, rmax
       do J=1,N
        do I=1,M
          do k=1,3
        if (iCV(I,J,K).eq.1) CV(I,J,K)=1.0
           enddo
         enddo
       enddo
c
c      Write to PPM File
c
       OPEN(UNIT=20,FILE='gamut.ppm',STATUS='UNKNOWN')
1      FORMAT(A10)
       WRITE(20,1) 'P3        '
       WRITE(20,1) '# gamut.ppm'
       WRITE(20,*) M,N
       WRITE(20,*) L
       DO J=1,N
        DO I=1,M
100      FORMAT(3(I4,2X))
     WRITE(20,100) (nint(255*CV(I,J,k)),k=1,3)
        ENDDO
       ENDDO
       STOP
       END
c ***********************************************************************
C   Should we wish to go out of gamut:
      SUBROUTINE XYZTORGB(xr,yr,zr,xg,yg,zg,xb,yb,zb,xc,yc,zc,r,g,b)
      IMPLICIT REAL*8 (a-h,o-z)
       r=(-xg*yc*zb+xc*yg*zb+xg*yb*zc-xb*yg*zc-xc*yb*zg+xb*yc*zg)/
     *   (+xr*yg*zb-xg*yr*zb-xr*yb*zg+xb*yr*zg+xg*yb*zr-xb*yg*zr)
      g=(+xr*yc*zb-xc*yr*zb-xr*yb*zc+xb*yr*zc+xc*yb*zr-xb*yc*zr)/
     *   (+xr*yg*zb-xg*yr*zb-xr*yb*zg+xb*yr*zg+xg*yb*zr-xb*yg*zr)
      b=(+xr*yg*zc-xg*yr*zc-xr*yc*zg+xc*yr*zg+xg*yc*zr-xc*yg*zr)/
     *   (+xr*yg*zb-xg*yr*zb-xr*yb*zg+xb*yr*zg+xg*yb*zr-xb*yg*zr)
      IF (R.LT.0.) R=0.
      IF (G.LT.0.) G=0.
      IF (B.LT.0.) B=0.
      IF (R.GT.1.) R=1.
      IF (G.GT.1.) G=1.
      IF (B.GT.1.) B=1.
     RETURN
      END
c ***********************************************************************
    real*8 function max(r,g,b)
      rmax = r
      if (g .gt. rmax) rmax = g
      if (b .gt. rmax) rmax = b
    end
c ***********************************************************************
```

This code is also given as
ExerciseAnswers/resources_exercises/chap4/makegamutppm.f

In C, the code looks as below.

http://www.fourmilab.ch/documents/specrend/ also has a C program version, `specrend.c`**, to**

output ascii values:

```
Built-in test program which displays the x, y, and Z and RGB
values for black body spectra from 1000 to 10000 degrees kelvin.
When run, this program should produce the following output:

Temperature       x      y      z       R     G     B
-----------    ------ ------ ------   ----- ----- -----
   1000 K      0.6528 0.3444 0.0028   1.000 0.007 0.000 (Approximation)
   1500 K      0.5857 0.3931 0.0212   1.000 0.126 0.000 (Approximation)
   2000 K      0.5267 0.4133 0.0600   1.000 0.234 0.010
   2500 K      0.4770 0.4137 0.1093   1.000 0.349 0.067
   3000 K      0.4369 0.4041 0.1590   1.000 0.454 0.151
   3500 K      0.4053 0.3907 0.2040   1.000 0.549 0.254
   4000 K      0.3805 0.3768 0.2428   1.000 0.635 0.370
   4500 K      0.3608 0.3636 0.2756   1.000 0.710 0.493
   5000 K      0.3451 0.3516 0.3032   1.000 0.778 0.620
   5500 K      0.3325 0.3411 0.3265   1.000 0.837 0.746
   6000 K      0.3221 0.3318 0.3461   1.000 0.890 0.869
   6500 K      0.3135 0.3237 0.3628   1.000 0.937 0.988
   7000 K      0.3064 0.3166 0.3770   0.907 0.888 1.000
   7500 K      0.3004 0.3103 0.3893   0.827 0.839 1.000
   8000 K      0.2952 0.3048 0.4000   0.762 0.800 1.000
   8500 K      0.2908 0.3000 0.4093   0.711 0.766 1.000
   9000 K      0.2869 0.2956 0.4174   0.668 0.738 1.000
   9500 K      0.2836 0.2918 0.4246   0.632 0.714 1.000
  10000 K      0.2807 0.2884 0.4310   0.602 0.693 1.000
```

```
/* makegamutppm.c
   Places an ascii  .ppm (portable pixmap) image of approximation
   of chromaticity diagram on stdout.
   Link the resulting object file with the math library:
   gcc makegamutppm.c -o makegamutppm -lm
   This code is an extension of a program in Fortran
   at link
   http://www.physics.sfasu.edu/astro/color/chromaticity.html
*/

#include <math.h>
#include <string.h>

/* Table of constant values */

/* Width and height and color depth of the ppm image; gamma */
#define N     300 /* square image */
#define LEVELS   255
#define GAMMA    1.0
#define NN3      N*N*3
#define NN       N*N
#define NPLUS1   N+1
#define NNNPLUS1 NN+NPLUS1

/* Builtin functions */
double over_pow();

/* Local functions */
double over_pow();
double transfersign();
double max3();


/* Main program */ main()
{
/* Initialized data */

/*      Chromaticity Coodinates (x and y) for wavelengths in 5 nm */
/*      increments from 380 nm to 780 nm.0*/
```

```
    static double xy81[164]    /* think of as [2][82] */
        = { .1741,.005,.174,.005,
       .1738,.0049,.1736,.0049,.1733,.0048,.173,.0048,.1726,.0048,.1721,
       .0048,.1714,.0051,.1703,.0058,.1689,.0069,.1669,.0086,.1644,.0109,
       .1611,.0138,.1566,.0177,.151,.0227,.144,.0297,.1355,.0399,.1241,
       .0578,.1096,.0868,.0913,.1327,.0687,.2007,.0454,.295,.0235,.4127,
       .0082,.5384,.0039,.6548,.0139,.7502,.0389,.812,.0743,.8338,.1142,
       .8262,.1547,.8059,.1929,.7816,.2296,.7543,.2658,.7243,.3016,.6923,
       .3373,.6589,.3731,.6245,.4087,.5896,.4441,.5547,.4788,.5202,.5125,
       .4866,.5448,.4544,.5752,.4242,.6029,.3965,.627,.3725,.6482,.3514,
       .6658,.334,.6801,.3197,.6915,.3083,.7006,.2993,.7079,.292,.714,
       .2859,.719,.2809,.723,.277,.726,.274,.7283,.2717,.73,.27,.7311,
       .2689,.732,.268,.7327,.2673,.7334,.2666,.734,.266,.7344,.2656,
       .7346,.2654,.7347,.2653,.7347,.2653,.7347,.2653,.7347,.2653,.7347,
       .2653,.7347,.2653,.7347,.2653,.7347,.2653,.7347,.2653,.7347,.2653,
       .7347,.2653,.7347,.2653,.7347,.2653,.7347,.2653,.7347,.2653,.7347,
       .2653,.7347,.2653 };


    /* locals */
    int i, j, k;
    int ii, jj, kk;
    int i1, i2, i3, i4, i5, i6, j1, j2;
    double rmax;
    double r, b, g;
    double s, x, y, slope;
    double s1, s2, t1, t2;
    double temp;
    static double rgb[NN3]; /* think of as [N][N][3] */
    double xb, yb, zb,
           xg, yg, zg,
           xr, yr, zr,
           xw, yw, zw,
           yy;

    double xx, zz;
    static int occup[NN]; /* think of as [N][N] */




    xy81[162] = xy81[0];
    xy81[163] = xy81[1];

/*      Chromaticity Coordinates for Red, Green, Blue phosphors */
/*      and White Point=D65 */
/*      SMPTE-C RGB: */
```

```
    xr = 0.63;
    yr = 0.34;
    xg = 0.31;
    yg = 0.595;
    xb = 0.155;
    yb = 0.07;
    xw = 0.312713;
    yw = 0.329016;
    zr = 1.0- (xr + yr);
    zg = 1.0- (xg + yg);
    zb = 1.0- (xb + yb);
    zw = 1.0- (xw + yw);

/*      Draw spectrum locus.    */

    for (ii = 1; ii <= N; ++ii) {
   for (jj = 1; jj <= N; ++jj) {
       for (kk = 1; kk <= 3; ++kk) {
       occup[ii + jj * N - NPLUS1] = 0;
       rgb[ii + (jj + kk * N) * N - NNNPLUS1] = 0.0;
       }
   }
    }
    j1 = 0;
    j2 = 0;
    for (j = 1; j <= 81; ++j) {
   s1 = N* xy81[(j << 1) - 2];
   s2 = N* xy81[(j + 1 << 1) - 2];
   t1 = N* (1.0- xy81[(j << 1) - 1]);
   t2 = N* (1.0- xy81[((j + 1) << 1) - 1]);
   if ((s2 - s1) != 0.0) {
       slope = (t2 - t1) / (s2 - s1);
   } else {
       s2 += 1e-5;
       slope = (t2 - t1) / (s2 - s1);
   }
   i1 = (int) s1;
   i2 = (int) s2;
   i3 = i2 - i1;
   i5 = transfersign(1.0, i3);
        ii = i1;
```

```
   while (i5 < 0 ? ii >= i2 : ii <= i2)
       {
       s = (double) ii;
       j1 = j2;
       j2 = (int) (t1 + slope * (s - s1));
       if ((j1 != 0) && (j2 != 0)) {
       i6 = j2 - j1;
       i4 = transfersign(1.0, i6);
       jj = j1;
       while (i4 < 0 ? jj >= j2 : jj <= j2)
           {
               occup[ii + jj * N - NPLUS1] = 1;
           jj += i4;
       }
       }
       ii += i5;

   }
    }

/*      Calculate RGB Values for x and y coordinates */

    rmax = 0.0;
    for (j = 1; j <= N; ++j) {
   for (i = 1; i <= N; ++i) {
       x = i * 1.0/ N;
       y = (N - j) * 1.0/ N + 1e-5;
       yy = 1.0;
       xx = x * yy / y;
       zz = (1.0- x - y) * yy / y;
/* smpte-c */
       r = xx * 3.5058 - yy * 1.7397 - zz * .544;
       g = -(xx * 1.069) + yy * 1.9778 + zz * .0352;
       b = xx * .0563 - yy * .197 + zz * 1.0501;

       if (! (occup[i + j * N - NPLUS1]) ) {
       if ((r < 0.0) || (g < 0.0) || (b < 0.0)) {
           r = 0.0;
           g = 0.0;
           b = 0.0;
       } else {
           r = over_pow(r, GAMMA);
           g = over_pow(g, GAMMA);
           b = over_pow(b, GAMMA);
```

```
/*                 Have same chromaticity if rescale RGB,
                   so brighten: */
           rmax = max3(r,g,b);
           r /= rmax;
           g /= rmax;
           b /= rmax;
         } /* else */
       rgb[i + (j + N) * N - NNNPLUS1] = r;
       rgb[i + (j + 2*N) * N - NNNPLUS1] = g;
       rgb[i + (j + 3*N) * N - NNNPLUS1] = b;
       } /* if */
   } /* for i */
    } /* for j */
/*      white point: draw a plus sign */
    i = (int) (xw * N);
    j = (int) (N - (yw - 1.0e-5) * N);
    i2 = i + 5;
    for (ii = i - 5; ii <= i2; ++ii) {
   rgb[ii + (j + N) * N - NNNPLUS1] = 0.0;
   rgb[ii + (j + 2*N) * N - NNNPLUS1] = 0.0;
   rgb[ii + (j + 3*N) * N - NNNPLUS1] = 0.0;
    }
    i2 = j + 5;
    for (jj = j - 5; jj <= i2; ++jj) {
   rgb[i + (jj + N) * N - NNNPLUS1] = 0.0;
   rgb[i + (jj + 2*N) * N - NNNPLUS1] = 0.0;
   rgb[i + (jj + 3*N) * N - NNNPLUS1] = 0.0;
    }

    for (j = 1; j <= N; ++j) {
   for (i = 1; i <= N; ++i) {
       for (k = 1; k <= 3; ++k) {
       if (occup[i + j * N - NPLUS1] == 1) {
           rgb[i + (j + k * N) * N - NNNPLUS1] = 1.0;
       }
       }
   }
    }
```

```
/*      Write ascii PPM file to stdout */

    printf("P3        \n");
    printf("# gamut.ppm\n");
    printf("%d %d\n", N,N);
    printf("%d\n", LEVELS);

    for (j = 1; j <= N; ++j) {
   for (i = 1; i <= N; ++i) {
       for (k = 1; k <= 3; ++k) {
       temp = rgb[i + (j + k * N) * N - NNNPLUS1] ;
       temp = rgb[i + (j + k * N) * N - NNNPLUS1] * 255;
       i2 = (int)(temp);
       printf("%d ", i2);
       }
       printf("\n");
   }
    }
} /* end of main */
```

```
/*************************************************/
double max3(r, g, b)
double r, g, b;
{
    /* Local variable */
    double rmax;

    rmax = r;
    if (g > rmax) {
   rmax = g;
    }
    if (b > rmax) {
   rmax = b;
    }
    return rmax;
} /*  end of max3 */

/*************************************************/
/* does transfer of sign, |a1|* sign(a2) */
double transfersign(a1,a2)
double a1,a2;
{
   if (a2>=0)
      return((double)(fabs(a1)));
   else
      return((double)(-1.0*fabs(a1)));
}

/*************************************************/
double over_pow(x,p)
double x,p;
  {
     if ( (x < 0.0) && ((p*0.5) - (int)(p*0.5) != 0.0) )
   {
   return(-pow(-x,p)) ;
   }
     else
   return(pow(x,p)) ;
  } /* end of over_pow */
```

This code is also given as
ExerciseAnswers/resources_exercises/chap4/makegamutppm.c

10. Hue is the color, independent of brightness and how much pure white has been added to it. We can make a *simple* definition of hue as the set of ratios R:G:B. Suppose a color (i.e., an RGB) is divided by 2.0, so that the RGB triple now has values 0.5 times its former values. Explain, using numerical values:

    (a) If gamma correction is applied after the division by 2.0 and before the color is stored, does the darker RGB have the same hue as the original, in the sense of having the same ratios R:G:B of *light* emanating from the CRT display device? (We're not discussing any psychophysical effects that change our perception — here we're just worried about the machine itself).

(b) If gamma correction is *not* applied, does the second RGB have the same hue as the first, when displayed?

(c) For what color triples is the hue always unchanged?

Answer:
(a) 2 With gamma correction, RGB is stored as (RGB)∧(1/gamma) and (RGB/2) is stored as (RGB/2)∧(1/gamma). After the CRT gamma takes effect, color∧gamma, the gamma-correction power law is reversed, and we're back to RGB and RGB/2, so the hue does not change.

(b) 2 But if there is no gamma-correction, then RGB results in light (RGB)∧gamma, and RGB/2 is viewed as (RGB/2)∧gamma, so we see a different hue. As an example, suppose RGB=1,1/2,1/4. Suppose gamma is 2.0. Then the color viewed is 1,1/4,1/16, which is a different hue.

(c) 1 Suppose RGB=1,1,1. Then the color viewed is also 1,1,1. And RGB/2=1/2*(1,1,1) is viewed as 1/4*(1,1,1) so the hue is unchanged (just darker). [Also, if RGB=(1,0,0), then RGB/2=1/2*(1,0,0) is displayed as 1/4*(1,0,0), which is the same hue. Any color with some equal entries, e.g. (1,1,0) will be the same hue, just darker. And any color with two "0" entries will also be the same hue, just darker.]

11. We wish to produce a graphic that is pleasing and easily readable. Suppose we make the background color `pink`. What color text font should we use to make the text most readable? Justify your answer.

Answer:

```
Pink is a mixture of white and red;  say that there are half of
each:  1 1 1
       1 0 0
       -----
       2 1 1 / 2  --> 1,.5,.5 = pink  (Chrom = .5, .25)

Then complementary color is (1,1,1)-pink = (0,.5,.5) (Chrom = 0,.5)
which is pale cyan.  [colorwh.frm]
```

12. To make matters simpler for eventual printing, we buy a camera equipped with CMY sensors, as opposed to RGB sensors (CMY cameras are in fact available).

(a) Draw spectral curves roughly depicting what such a camera's sensitivity to frequency might look like.

Answer:

(see plot Fig. 4.22)

(b) Could the output of a CMY camera be used to produce ordinary RGB pictures? How?

Answer:
Suppose we had C=G+B (i.e., with coefficients 1.0 times G and B), etc. Then C=R,G,B - R,0,0 = 1-R, and so R=1-C. So use R=1-C, G=1-M, B=1-Y.

13. Color inkjet printers use the CMY model. When the cyan ink color is sprayed onto a sheet of white paper,

(a) Why does it look cyan under daylight?

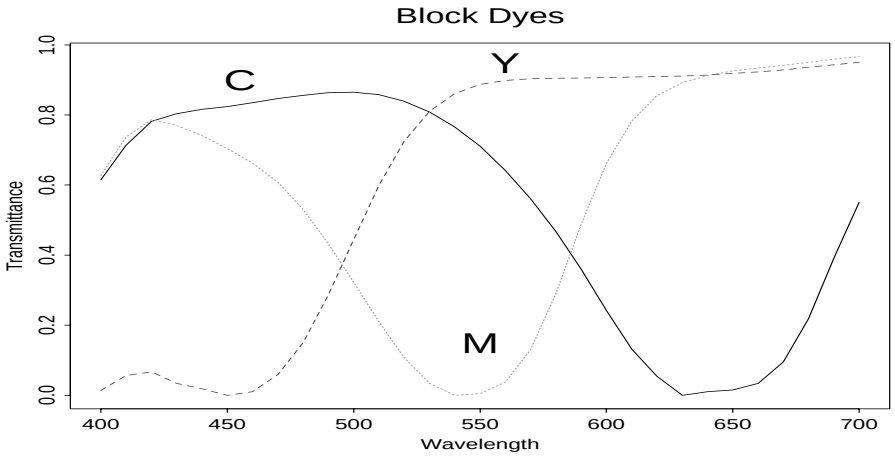


Fig. 4.22: CMY “Block” dye transmittance.

(b) What color would it appear under a blue light? Why?

Answer:
(i) RED from the daylight is absorbed (subtracted).
(ii) BLUE. The CYAN ink will not absorb BLUE, and BLUE is the only color in the light.

Chapter 5

Fundamental Concepts in Video

Exercises

1. NTSC video has 525 lines per frame and 63.6 μsec per line, with 20 lines per field of vertical retrace and 10.9 μsec horizontal retrace.

   (a) Where does the 63.6 μsec come from?
   Answer:

   1 / (525 lines/frame$\times$29.97 frame/sec) = 63.6$\times 10^{-6}$ sec/line

   (b) Which takes more time, horizontal retrace or vertical retrace? How much more time?
   Answer:

   horiz = 10.9$\times 10^{-6}$ sec,
   vertical is 20 line * 63.6 μsec = 1272 μsec = 1.272 msec, so
   vertical is 1272/10.9 = 117 times longer than horizontal.

2. Which do you think has less detectable flicker, PAL in Europe or NTSC is North America? Justify your conclusion.

   Answer:
   PAL could be better since more lines, but is worse because of fewer frames/sec.

3. Sometimes the signals for television are combined into fewer than all the parts required for TV transmission.

   (a) Altogether, how many and what are the signals used for studio broadcast TV?
   Answer:
   5
   R, G, B, audio, sync; can say "blanking" instead, too.

   (b) How many and what signals are used in S-Video? What does S-Video stand for?
   Answer:
   Luminance+chrominance = 2+audio+sync = 4
   Separated video

   (c) How many signals are actually broadcast for standard analog TV reception? What kind of video is that called?

Answer:
1
Composite

4. Show how the Q signal can be extracted from the NTSC chroma signal C (Eq. 5.1) during the demodulation process.

   Answer:

   To extract Q:

   (a) Multiply the signal C by $2\sin(F_{sc}t)$, i.e.,

$$\begin{aligned} C \cdot 2\sin(F_{sc}t) &= I \cdot 2\sin(F_{sc}t)\cos(F_{sc}t) + Q \cdot 2\sin^2(F_{sc}t) \\ &= I \cdot \sin(2F_{sc}t) + Q \cdot (1 - \cos(2F_{sc}t)) \\ &= Q + I \cdot \sin(2F_{sc}t) - Q \cdot \cos(2F_{sc}t). \end{aligned}$$

   (b) Apply a low-pass filter to obtain Q and discard the two higher frequency ($2F_{sc}$) terms.

5. One sometimes hears that the old Betamax format for videotape, which competed with VHS and lost, was actually a better format. How would such a statement be justified?

   Answer:
   Betamax has more samples per line: 500, as opposed to 240.

6. We don't see flicker on a workstation screen when displaying video at NTSC frame rate. Why do you think this might be?

   Answer:
   NTSC video is displayed at 30 frames per sec, so flicker is possibly present. Nonetheless, when video is displayed on a workstation screen the video buffer is read and then rendered on the screen at a much higher rate, typically the refresh rate — 60 to 90 Hz — so no flicker is perceived. (And in fact most display systems have double buffers, completely removing flicker: since main memory is much faster than video memory, keep a copy of the screen in main memory and then when we this buffer update is complete, the whole buffer is copied to the video buffer.)

7. Digital video uses *chroma subsampling*. What is the purpose of this? Why is it feasible?

   Answer:
   Human vision has less acuity in color vision than it has in black and white — one can distinguish close black lines more easily than colored lines, which soon are perceived just a mass without texture as the lines move close to each other. Therefore, it is acceptable perceptually to remove a good deal of color information. In analog, this is accomplished in broadcast TV by simply assigning a smaller frequency bandwidth to color than to black and white information. In digital, we "decimate" the color signal by subsampling (typically, averaging nearby pixels). The purpose is to have less information to transmit or store.

8. What are the most salient differences between ordinary TV and HDTV?

   Answer:
   More pixels, and aspect ratio of 16/9 rather than 4/3.
   What was the main impetus for the development of HDTV?

Immersion — "being there". Good for interactive systems and applications such as virtual reality.

9. What is the advantage of interlaced video? What are some of its problems?

   Answer:
   Positive: Reduce flicker. Negative: Introduces serrated edges to moving objects and flickers along horizontal edges.

10. One solution that removes the problems of interlaced video is to de-interlace it. Why can we not just overlay the two fields to obtain a de-interlaced image? Suggest some simple de-interlacing algorithms that retain information from both fields.

    Answer:
    The second field is captured at a later time than the first, creating a temporal shift between the odd and even lines of the image.
    The methods used to overcome this are basically two: non-motion compensated and motion compensated de-interlacing algorithms.

    The simplest non-motion compensated algorithm is called "Weave"; it performs linear interpolation between the fields to fill in a full, "progressive", frame. A defect with this method is that moving edges show up with significant serrated lines near them.

    A better algorithm is called "Bob": in this algorithm, one field is discarded and a a full frame is interpolated from a single field. This method generates no motion artifacts (but of course detail is reduced in the resulting progressive image).

    In a vertical-temporal (VT) de-interlacer, vertical detail is reduced for higher temporal frequencies. Other, non-linear, techniques are also used.

    Motion compensated de-interlacing performs inter-field motion compensation and then combines fields so as to maximize the vertical resolution of the image.

Chapter 6

Basics of Digital Audio

Exercises

1. My old Soundblaster card is an 8–bit card.

   (a) What is it 8 bits of?

   Answer:
   Quantization levels (not sampling frequency).

   (b) What is the best SQNR (Signal to Quantization Noise Ratio) it can achieve?

   Answer:

```
(a) Quantization levels (not sampling frequency)
(b) Best SQNR is 1 level out of 256 possible levels.
Calculate SQNR using largest value in dynamic range:
    SNR = 20 log_10 (255/2^0 )
       ~= 20 log 2^8
        = 20*8*log 2
       ~= 20*8* 0.3
    = 48 db  (actually, 48.16 db)
```

2. If a set of ear protectors reduces the noise level by 30 dB, how much do they reduce the intensity (the power)?

   Answer:
   A reduction in intensity of 1000.

3. A loss of audio output at both ends of the audible frequency range is inevitable, due to the frequency response function of an audio amplifier and the medium (e.g., tape).

   (a) If the output was 1 volt for frequencies at midrange, what is the output voltage after a loss of -3 dB at 18 kHz?

   (b) To compensate for the loss, a listener can adjust the gain (and hence the output) on an equalizer at different frequencies. If the loss remains -3 dB and a gain through the equalizer is 6 dB at 18 kHz, what is the output voltage now? *Hint:* Assume $\log_{10} 2 = 0.3$.

Answer:
(a) $20 \log \frac{V}{1} = -3$**;** $2 \log V = -0.3$**;** $2 \log V = -\log 2$**;** $\log(V^2) = -\log 2$**;** $V = \frac{1}{\sqrt{2}} = 0.7$ **volts**
(b) -3 + 6 = 3 dB; $V = \sqrt{2} = 1.4$ **volts**

4. Suppose the sampling frequency is 1.5 times the true frequency. What is the alias frequency?

   Answer:
   0.5 times the True Frequency.

5. In a crowded room, we can still pick out and understand a nearby speaker's voice, notwithstanding the fact that general noise levels may be high. This is known as the *cocktail-party effect*. The way it operates is that our hearing can localize a sound source by taking advantage of the difference in phase between the two signals entering our left and right ears (*binaural auditory perception*). In mono, we could not hear our neighbor's conversation well if the noise level were at all high. State how you think a karaoke machine works. *Hint:* The mix for commercial music recordings is such that the "pan" parameter is different going to the left and right channels for each instrument. That is, for an instrument, either the left or right channel is emphasized. How would the singer's track timing have to be recorded to make it easy to subtract the sound of the singer (which is typically done)?

   Answer:
   For the singer, left and right is always mixed with the exact same pan. This information can be used to subtract out the sound of the singer. To do so, replace the left channel by the difference between the left and the right, and boost the maximum amplitude; and similarly for the right channel.

6. The *dynamic range* of a signal V is the ratio of the maximum to the minimum absolute value, expressed in decibels. The dynamic range expected in a signal is to some extent an expression of the signal quality. It also dictates the number of bits per sample needed to reduce the quantization noise to an acceptable level. For example, we may want to reduce the noise to at least an order of magnitude below V_{min}. Suppose the dynamic range for a signal is 60 dB. Can we use 10 bits for this signal? Can we use 16 bits?

   Answer:
   The range is mapped to $-2^{(N-1)} \ldots 2^{(N-1)} - 1$**.** V_{max} **is mapped to top value,** $\sim 2^{(N-1)}$**. In fact, whole range** V_{max} **down to** $(V_{max} - q/2)$ **is mapped to that, where** q **is the quantization interval. The largest negative signal,** $-V_{max}$ **is mapped to** $-2^{(N-1)}$**. Therefore** $q = (2 * V_{max})/(2^N)$**, since there are** 2^N **intervals.**

   The dynamic range is V_{max}/V_{min}**, where** V_{min} **is the smallest *positive* voltage we can see that is not masked by the noise. Since the dynamic range is 60 dB, we have** $20 \log_{10}(V_max/V_min) = 60$ **so** $V_{min} = V_{max}/1000$**.**
   At 10 bits, the quantization noise, equal to $q/2$**==half a quantization interval** q**, is** $q/2 = (2 * V_{max}/2^N)/2 = V_{max}/(2^{10})$**, or in other words** $V_{max}/1024$**. So this is not sufficient intensity resolution.**
   At 16 bits, the noise is $V_{max}/(2^{16}) = V_{max}/(64*1024)$**, which is more than an order of magnitude smaller than** V_{min} **so is fine.**

7. Suppose the dynamic range of speech in telephony implies a ratio V_{max}/V_{min} of about 256. Using uniform quantization, how many bits should we use to encode speech to make the quantization noise at least an order of magnitude less than the smallest detectable telephonic sound?

   Answer:
   $V_{min} = V_{max}/256$**.**

The quantization noise is $V_{max}/2^n$, if we use n bits. Therefore to get quantization noise about a factor of 16 below the minimum sound, we need 12 bits.

8. *Perceptual nonuniformity* is a general term for describing the nonlinearity of human perception. That is, when a certain parameter of an audio signal varies, humans do not necessarily perceive the difference in proportion to the amount of change.

   (a) Briefly describe at least two types of perceptual nonuniformities in human auditory perception.

   (b) Which one of them does A-law (or μ-law) attempt to approximate? Why could it improve quantization?

   Answer:
   (a):
   (1) Logarithmic response to magnitude,
   (2) different sensitivity to different frequencies,
   (b): A-law (or μ-law) approximates the non-linear response to magnitude. It makes better use of the limited number of bits available for each quantized data.

9. Draw a diagram showing a sinusoid at 5.5 kHz and sampling at 8 kHz (show eight intervals between samples in your plot). Draw the alias at 2.5 kHz and show that in the eight sample intervals, exactly 5.5 cycles of the true signal fit into 2.5 cycles of the alias signal.

   Answer:

```
% matlab script
% time = 1 msec only.
truesamplingfreq = 1000*5.5; % 5.5 MHz
freqin1msec = 5.5; % 5.5 cycles per msec.
aliassamplingfreq = 1000*2.5; % 2.5 MHz
aliasfreqin1msec = 2.5; % 2.5 cycles per msec.
drawingpoints = 1000; % just for making smooth figure.
time = (0:drawingpoints)/drawingpoints;
% define a signal with 5.5x10^3 cps.
signal = sin(freqin1msec*2*pi*time);
% define an alias signal with 2.5x10^3 cps.
alias = -sin(aliasfreqin1msec*2*pi*time);
% And how did we get this alias from sampling at 8kHz?--
% We undersample at 8kHz==> just 8 samples in 1msec.
undersamplinginterval = max(time)/8.0;
undersamplingtimes = (0:undersamplinginterval:max(time));
undersampled = sin(freqin1msec*2*pi*undersamplingtimes);
%
plot(time,signal,'g','LineWidth',2);
hold on
plot(time,alias,'r--');
%
plot(undersamplingtimes,undersampled, 'ob');
xlabel('Time (msec)');
```

```
ylabel('Signal');
hold off
print -depsc /figs/chap6/alias.eps
```

10. Suppose a signal contains tones at 1, 10, and 21 kHz and is sampled at the rate 12 kHz (and then processed with an antialiasing filter limiting output to 6 kHz). What tones are included in the output? *Hint:* Most of the output consists of aliasing.

    Answer:
    1 kHz, 12-10=2 kHz, and 2*12-21=3 kHz tones are present.

11. (a) Can a single MIDI message produce more than one note sounding?

    Answer:
    No.

    (b) Is it possible for more than one note to sound at once on a particular instrument? If so, how is it done in MIDI?

    Answer:
    Yes — use two NoteOn messages for one channel before the NoteOff message is sent.

    (c) Is the Program Change MIDI message a Channel Message? What does this message accomplish? Based on the Program Change message, how many different instruments are there in General MIDI? Why?

    Answer:
    Yes.
    Replaces patch for a channel.
    128, since has one data byte, which must be in 0..127.

    (d) In general, what are the two main kinds of MIDI messages? In terms of data, what is the main difference between the two types of messages? Within those two categories, list the different subtypes.

    Answer:
    Channel Messages and System Messages.
    Channel voice messages, Channel mode messages, System real-time messages, System common messages, System exclusive messages.
    Channel messages have a status byte with leading most-significant-bit set, and 4 bits of channel information; System messages have the 4 MSBs set.

12. (a) Give an example (in English, not hex) of a MIDI voice message.

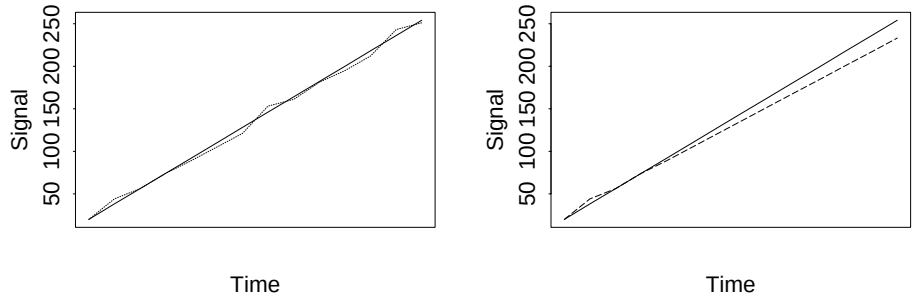
    Answer:
    NoteOn

    (b) Describe the parts of the "assembler" statement for the message.

    Answer:
    (1) opcode=Note on; (2) data = note, or key, number; (3) data = "velocity"==loudness.

    (c) What does a Program Change message do? Suppose Program change is hex "&HC1." What does the instruction "&HC103" do?

    Answer:
    Changes the patch to #4 on channel 2.



(a) (b)

Fig. 6.19: (a) DPCM reconstructed signal (dotted line) tracks the input signal (solid line). (b) DPCM reconstructed signal (dashed line) steers farther and farther from the input signal (solid line).

13. In PCM, what is the *delay*, assuming 8 kHz sampling? Generally, delay is the time penalty associated with any algorithm due to sampling, processing, and analysis.

    Answer:
    Since there is no processing associated with PCM, the delay is simply the time interval between two samples, and at 8 kHz, this is 0.125 msec.

14. (a) Suppose we use a predictor as follows:

$$\begin{aligned} \hat{f}_n &= \text{trunc}\left[\tfrac{1}{2}(\tilde{f}_{n-1} + \tilde{f}_{n-2})\right] \\ e_n &= f_n - \hat{f}_n \end{aligned} \tag{6.25}$$

    Also, suppose we adopt the quantizer Equation (6.20). If the input signal has values as follows:

    20 38 56 74 92 110 128 146 164 182 200 218 236 254

    show that the output from a DPCM coder (without entropy coding) is as follows:

    20 44 56 74 89 105 121 153 161 181 195 212 243 251

    Figure 6.19(a) shows how the quantized reconstructed signal tracks the input signal. As a programming project, write a small piece of code to verify your results.

    (b) Suppose by mistake on the coder side we inadvertently use the predictor for *lossless coding*, Equation (6.14), using original values f_n instead of quantized ones, $\tilde{f}_n$. Show that on the decoder side we end up with reconstructed signal values as follows:

    20 44 56 74 89 105 121 137 153 169 185 201 217 233

    so that the error gets progressively worse.

    Figure 6.19(b) shows how this appears: the reconstructed signal gets progressively worse. Modify your code from above to verify this statement.

    Answer:

```
% dcpm.m -- matlab script-- included in figs/chap6
%Let the predictor be
```

```
%s_nˆ=fix((s_n-1+s_n-2)/2)
%and let the quantizer be
%Q(s(n))  = 16*fix(e/16) + 8 % NO  % NO  % NO  % NO  % NO
%%%%%%
errors = (-255):255; %set up for errors % length==511
% Quantizer:
errorstilde =16*fix((255+errors)/16) - 256 + 8;
% ==which errors go to which levels.
plot(errors,errorstilde); % staircase
%%%%%%
load 'figs/chap6/signal.dat' signal -ascii
% 20;38;56;74;92;110;128 146;164;182;200;218;236;254
signal = [signal(1), signal]'; % append an extra first element
ss = size(signal);
errorunquantized = zeros(ss,1);  % declare
signaltilde = errorunquantized; % declare
signalpredicted = errorunquantized; % declare
%% first and second elements are exact:
signalpredicted(1:2) = signal(1:2);
errorunquantized(1:2) = 0;
errorquantized(1:2) = 0;
signaltilde(1:2) = signal(1:2);
%%
for i = 3:ss
  signalpredicted(i) = fix((signaltilde(i-1)+...
                       signaltilde(i-2))/2);
  % in (-255):255
  errorunquantized(i) = signal(i) - signalpredicted(i);
  errorquantized(i) = 16*fix((255+errorunquantized(i))/16)...
                      - 256 + 8;
  signaltilde(i) = signalpredicted(i) + errorquantized(i);
end

% decode: what's received is  errorquantized(i)
for i = 3:ss
  signalpredicted(i) = fix((signaltilde(i-1)+...
                       signaltilde(i-2))/2);
  signaltilde(i) = signalpredicted(i) + errorquantized(i);
end
% signal =    20  20  38  56  74  92 110 128 146 164 182 200
%              218 236 254
% signaltilde = 20  20  44  56  74  89 105 121 153 161 181
%              195 212 243 251
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Now let's do it wrong:
signaltildewrong = signaltilde;
signalpredictedwrong = signalpredicted;
```

```
errorunquantizedwrong = errorunquantized;
errorquantizedwrong = errorquantized;
for i = 3:ss
  signalpredictedwrong(i) = fix((signal(i-1)+signal(i-2))/2);
  % in (-255):255
  errorunquantizedwrong(i) = signal(i) - signalpredictedwrong(i);
  errorquantizedwrong(i) = 16*fix((255+...
                          errorunquantizedwrong(i))/16) - 256 + 8;
  signaltildewrong(i) = signalpredictedwrong(i) +...
                        errorquantizedwrong(i);
end
% signaltildewrong = 20  20  44  53  71  89 107 125 143 161
%                    179 197 215 233 251
% errorquantizedwrong = 0  0 24 24 24 24 24 24 24 24 24 24
%                       24 24 24
% decode:
for i = 3:ss
  signalpredictedwrong(i) = fix((signaltildewrong(i-1)+...
                             signaltildewrong(i-2))/2);
  signaltildewrong(i) = signalpredictedwrong(i) +...
                        errorquantizedwrong(i);
end
% Now signaltildewrong = 20  20  44  56  74  89 105 121 137 153...
%                       169 185 201 217 233
% Gets progressively lower than the correct line.

plot(signal(2:end));
xlabel('Time');
ylabel('Signal');
hold on
plot(signaltilde(2:end),'--');
hold off

plot(signal(2:end));
xlabel('Time');
ylabel('Signal');
hold on
plot(signaltildewrong(2:end),'--');
hold off
```

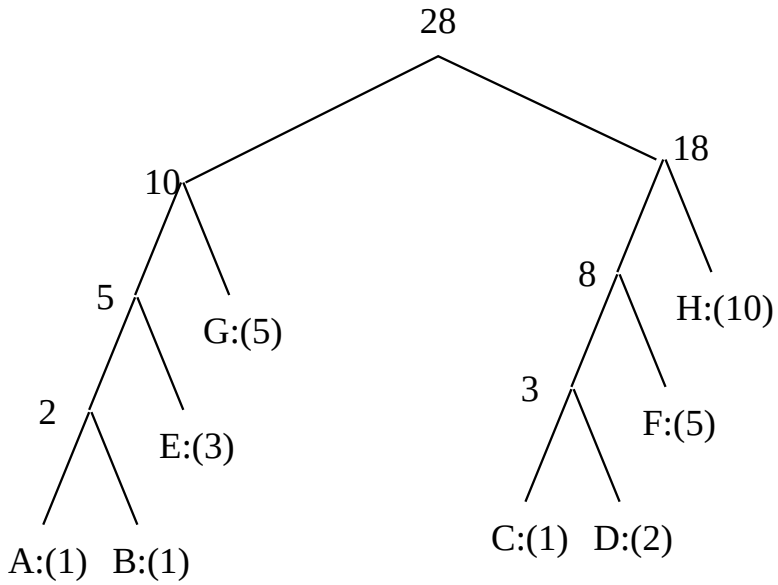
Chapter 7

Lossless Compression Algorithms

Exercises

1. Suppose eight characters have a distribution A:(1), B:(1), C:(1), D:(2), E:(3), F:(5), G:(5), H:(10). Draw a Huffman tree for this distribution. (Because the algorithm may group subtrees with equal probability in a different order, your answer is not strictly unique.)

   Answer:



2. (a) What is the entropy (η) of the image below, where numbers (0, 20, 50, 99) denote the gray-level intensities?

$$\begin{array}{cccccccc}
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\
20 & 20 & 20 & 20 & 20 & 20 & 20 & 20 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 50 & 50 & 50 & 50 & 0 & 0 \\
0 & 0 & 50 & 50 & 50 & 50 & 0 & 0 \\
0 & 0 & 50 & 50 & 50 & 50 & 0 & 0 \\
0 & 0 & 50 & 50 & 50 & 50 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

(b) Show step by step how to construct the Huffman tree to encode the above four intensity values in this image. Show the resulting code for each intensity value.

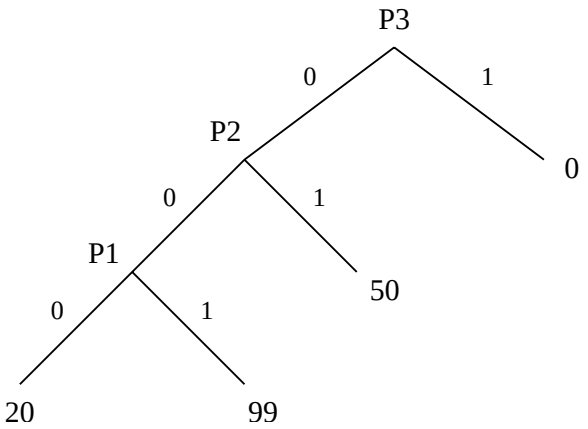
(c) What is the average number of bits needed for each pixel, using your Huffman code? How does it compare to η?

Answer:

(a) $P_{20} = P_{99} = 1/8$, $P_{50} = 1/4$, $P_0 = 1/2$.

$$\eta = 2 \times \frac{1}{8}\log_2 8 + \frac{1}{4}\log_2 4 + \frac{1}{2}\log_2 2 = \frac{3}{4} + \frac{1}{2} + \frac{1}{2} = 1.75$$

(b) Only the final tree is shown below. Resulting code: 0: "1", 50: "01", 20: "000", 99: "001"



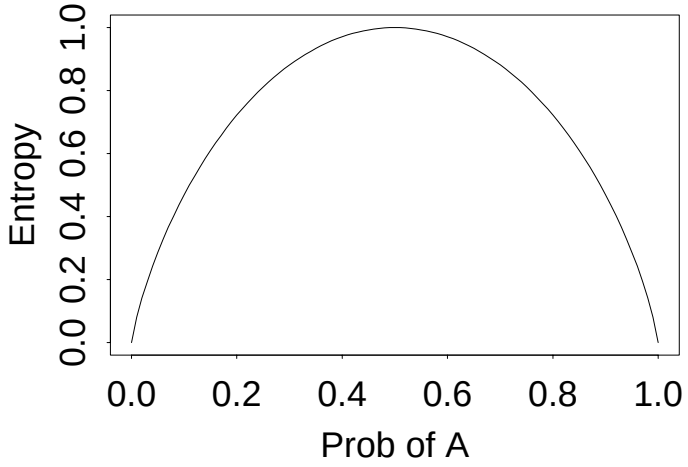
(c) Average number of bits = $0.5 \times 1 + 0.25 \times 2 + 2 \times 0.125 \times 3 = 1.75$.

This happens to be identical to η — it only happens when all probabilities are 2^{-k} where k is an integer. Otherwise, this number will be larger than η.

3. Consider an alphabet with two symbols A, B, with probability $P(A) = x$ and $P(B) = 1 - x$.

(a) Plot the entropy as a function of x. You might want to use $log_2(3) = 1.6$, $log_2(7) = 2.8$.

Answer:



If $x = 1/2$**,** $\eta = 1/2 + 1/2 = 1$**.**
If $x = 1/4$**,** $\eta = 1/4 \times 2 + 3/4 \times (log_2(4) - log_2(3))$
$= 1/2 + 3/4 * (2 - 1.6)$ = $1/2 + 0.3$ = **.8**
If $x = 3/4$**, same.**
If $x = 1/8$**,** $\eta = 1/8 \times 3 \ + \ 7/8 \times log_2(8/7)$
=$3/8 + 7/8 \times (3 - log_2(7))$
=$3/8 + 7/8 \times (3 - 2.8)$
=$3/8 + 7/8 \times (.2)$
=$.375 + .175$ = $.55$
If $x = 0$**, do not count that symbol;** $\rightarrow \ \eta = 1 \times log_2(1)$ = **0**

```
%matlab script:
x = (0:0.01:1)';
ss = size(x,1);
x(1) = 0.000001;
x(ss) = 1.0-0.000001;
y = enf(x);
plot(x,y);
xlabel('Prob of A');
ylabel('Entropy');
print -depsc entropyplot.eps

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function theentropy = enf(x)
  theentropy = ( x.*log(1.0./x) + (1-x).*log(1.0./(1-x)) )...
               /log(2);
```

(b) Discuss why it must be the case that if the probability of the two symbols are $1/2+\epsilon$ and $1/2-\epsilon$, with small ϵ, the entropy is less than the maximum.

Answer:
$H = \sum_i p_i lg(1/p_i)$ **and** $\sum_i p_i = 1$**, so that if all probabilities are equal, then** $p_i = 1/N$**,** $H = \sum_i 1/N lg(N) = lg(N)$**. E.g., if N=256, H=8.**

Now if probabilities not all equal, then suppose e.g. that one is a little smaller than 1/N, and one is a little bigger. The small probability has a larger lg(1/probability) than the one for even distribution; and the large probability has a smaller lg(1/probability) than the one for even distribution. But since for $x < 1lg(x)$ changes faster than x [the derivative of x is 1.0, and that for lg(1/x) is -1/x/ln(2)=-x/0.69=-1.4427x], -1/x/ln(2)=-1.4427/x], the smaller probability has more of an effect, and the result is that the overall entropy is less.

[If we want to be formal, then suppose we have just 2 symbols, one with probability $1/2+\epsilon$, and the other with probability $1/2-\epsilon$. Forming a Taylor series,
$lg(p1) \simeq -1 + 2/ln(2) * \epsilon - 2/ln(2) * \epsilon^2$**;**
$lg(p2) \simeq -1 - 2/ln(2) * \epsilon - 2/ln(2) * \epsilon^2$**;**
so $H \simeq -(1/2+\epsilon) * lg(p1) - (1/2-\epsilon) * lg(p2)$ **equals** $1 - 2 * \epsilon^2/ln(2)$ **].**

(c) Generalize the above result by showing that, for a source generating N symbols, the entropy is maximum when the symbols are all equiprobable.

Answer:
for $i = 1..N$**,** $p_i = 1/N$ **if equi-probable.**
$H = \sum_{i=1}^{N} -1/N lg(1/N) = lg(N)$**.**
Consider 2 symbols, as above: equi-probable gives $H = 1$**, and not equi-probable reduces** H**. By induction, result follows.**

(d) As a small programming project, write code to verify the conclusions above.

Answer:

```
# Maple script:
# Suppose that have alphabet of 2 symbols,
# with prob's 1/2+eps,1/2-eps:
p1 := 1/2+eps;
p2 := 1/2-eps;
lp1 := log(p1)/log(2);
lp2 := log(p2)/log(2);
taylor(lp1,eps,3); # -1 + 2/ln(2) *eps - 2/ln(2)*epsˆ2;
taylor(lp2,eps,3); # -1 - 2/ln(2) *eps - 2/ln(2)*epsˆ2;
lp1 := -1 + 2/ln(2) *eps - 2/ln(2)*epsˆ2;
lp2 := -1 - 2/ln(2) *eps - 2/ln(2)*epsˆ2;
ans := - p1*lp1 - p2*lp2;
simplify(ans); # 1 - 2*epsˆ2/ln(2)
```

4. *Extended Huffman Coding* assigns one codeword to each group of k symbols. Why is $average(l)$ (the average number of bits for each symbol) still no less than the entropy η as indicated in Eq. (7.7)?

Answer:

Eq. 7.6 shows $\eta < \bar{l} < \eta + 1$ **for Huffman coding. If we simply treat each of the group symbols as a new symbol in an ordinary Huffman coding, and use** $\bar{l}^{(k)}$ **and** $\eta^{(k)}$ **as the average bit length for each new symbol and the new entropy, respectively, we have**

$$\eta^{(k)} \leq \bar{l}^{(k)} < \eta^{(k)} + 1$$

Since $\bar{l}^{(k)}$ is the average number of bits need for k symbols, $\bar{l}^{(k)} = k \cdot \bar{l}$. It follows,

$$\frac{\eta^{(k)}}{k} \leq \bar{l} < \frac{\eta^{(k)}}{k} + \frac{1}{k}$$

It can be proven (see Sayood [2], p.50 for proof) that $\eta^{(k)} = k \cdot \eta$. Therefore, for extended Huffman coding

$$\eta \leq \bar{l} < \eta + \frac{1}{k}.$$

— that is, the average number of bits for each symbol is still no less than the entropy η.

5. Arithmetic Coding and Huffman Coding are two popular lossless compression methods.

   (a) What are the advantages and disadvantages of Arithmetic Coding as compared to Huffman Coding?

   Answer:

   The main advantage of Arithmetic Coding over Huffman Coding is that whereas the minimum code length for a symbol in Huffman Coding is 1, since we create a binary tree with 0 or 1 attached to each branch, in Arithmetic Coding the number of bits per symbol can be fractional.

   (b) Suppose the alphabet is $[A, B, C]$, and the known probability distribution is $P_A = 0.5, P_B = 0.4, P_C = 0.1$. For simplicity, let's also assume that both encoder and decoder know that the length of the messages is always 3, so there is no need for a terminator.

      i. How many bits are needed to encode the message BBB by Huffman coding?
      Answer:
      6 bits. Huffman Code: A - 0, B - 10, C - 11; or A - 1, B - 00, C - 01.

      ii. How many bits are needed to encode the message BBB by arithmetic coding?
      Answer:

Symbol	low	high	range
	0	1.0	1.0
B	0.5	0.9	0.4
B	0.7	0.86	0.16
B	0.78	0.844	0.064

      4 bits. Binary codeword is 0.1101, which is 0.8125.

6. (a) What are the advantages of Adaptive Huffman Coding compared to the original Huffman Coding algorithm?

   (b) Assume that the Adaptive Huffman Coding is used to code an information source S with a vocabulary of four letters (a, b, c, d). Before any transmission, the initial coding is a = 00, b = 01, c = 10, d = 11. As in the example illustrated in Fig. 7.7, a special symbol NEW will be sent before any letter if it is to be sent the first time.

   Fig. 7.11 is the Adaptive Huffman Tree after sending letters **aabb**.

   After that, the additional bitstream received by the decoder for the next few letters is 01010010101.

      i. What are the additional letters received?

      ii. Draw the adaptive Huffman trees after each of the additional letters is received.

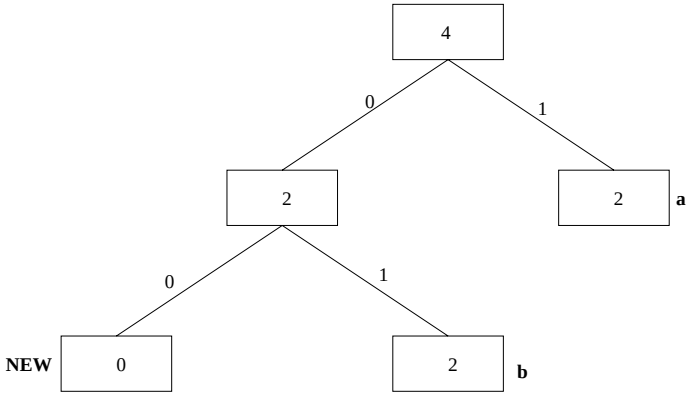


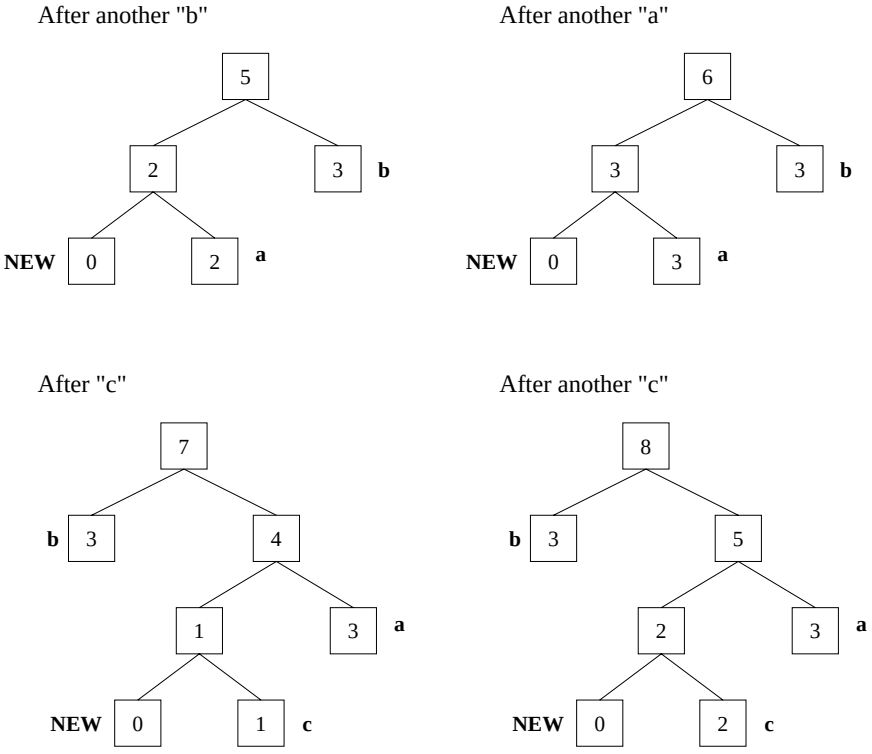
Fig. 7.11: Adaptive Huffman Tree.

Answer:

(a) Like any other adaptive compression algorithms, it is more dynamic, therefore offers better compression and works even when prior statistics of the data distribution is unavailable as it is in most multimedia applications. It also saves the overhead since no symbol table needs to be transmitted.

(b) (i) The additional letters received are "b (01) a (01) c (00 10) c (101)".

(ii) The trees are as below.



7. Compare the rate of adaptation of adaptive Huffman coding and adaptive arithmetic coding (see the textbbook web site for the latter). What prevents each method from adapting to quick changes in source statistics?

Answer:

Both methods would have a similar rate of adaptation since they both use symbol occurrences as estimates of symbol probability. The difference between the two algorithms is the way these symbol occurrences are utilized. In the adaptive Huffman case, the symbol occurrences are used as weights on each node to enforce the sibling property while adaptive arithmetic coding uses them to form cumulative frequency table. In any case, any change in the source statistics is reflected right away in both algorithms.

What prevents both algorithms to adapt to quick changes in input statistics is that symbols must occur enough times according to their probablity in order for the Huffman tree and the cumulative frequency table to emerge into a form that reflects the intended probabilities. Thus, if there is a large number of symbols that occurred many many times already, any quick changes in the input statistics will not be adapted very much for both algorithms. For adaptive arithmetic coding, this can be resolved by renormalizing the cumulative frequency table from time to time. Similarly, for adaptive Huffman coding, the Huffman tree can be purged to increase the adaptiveness.

8. Consider the dictionary-based LZW compression algorithm. Suppose the alphabet is the set of symbols {0,1}. Show the dictionary (symbol sets plus associated codes) and output for LZW compression of the input

0 1 1 0 0 1 1

Answer:
With input 0 1 1 0 0 1 1, we have

```
                        DICTIONARY
w   k  wk | output | index   symbol
-   -  -- | ------ | -----   ------
NIL 0  0  |        |
0   1  01 | 0      | 2       01
1   1  11 | 1      | 3       11
1   0  10 | 1      | 4       10
0   0  00 | 0      | 5       00
0   1  01 |        |
01  1  011| 2      | 6       011
1        |        |
(Students don't need to finish outputing
for the final input value of 1 since that
wasn't made clear in the algorithm.)
```

9. Implement Huffman coding, adaptive Huffman, arithmetic coding, and the LZW coding algorithms using your favorite programming language. Generate at least three types of statistically different artificial data sources to test your implementation of these algorithms. Compare and comment on each algorithm's performance in terms of compression ratio for each type of data source.

Answer:

To show the difference between these algorithms, the students can generate a uniformly distributed source, a normally distributed source, and a Markov source (source with memory). It is expected that arithmetic coding to perform well for the first two types, while the LZW algorithm should perform well for the last kind. The students should also note how well the input statistics is adapted in the adaptive Huffman algorithm.

Chapter 8

Lossy Compression Algorithms

Exercises

1. Assume we have an unbounded source we wish to quantize using an M-bit midtread uniform quantizer. Derive an expression for the total distortion if the step size is 1.

   Answer:
   The total distortion can be divided into two components: the granular distortion and overload distortion. Let $k = \frac{2^M}{2}$. Since we have an M bit midtread quantizer, the number of reconstruction levels are $k - 1$. Since two reconstruction values are allocated for the overload regions, there are $k - 3$ reconstruction levels in the granular region. Therefore, we have the following expression for the total distortion.

$$\begin{aligned} D &= D_g + D_o \\ &= \left(2 \sum_{i=1}^{\frac{k}{2}-2} \int_{i-0.5}^{i+0.5} (x-i)^2 f_X(x) dx + \int_{-0.5}^{0.5} x^2 f_X(x) dx \right) + \\ &\quad \left(2 \int_{\frac{k}{2}-2}^{\infty} (x - (\frac{k}{2} - 2 + 0.5))^2 f_X(x) dx \right) \end{aligned}$$

2. Suppose the domain of a uniform quantizer is $[-b_M, b_M]$. We define the loading fraction as

$$\gamma = \frac{b_M}{\sigma}$$

   where σ is the standard deviation of the source. Write a simple program to quantize a Gaussian distributed source having zero mean and unit variance using a 4-bit uniform quantizer. Plot the SNR against the loading fraction and estimate the optimal step size that incurs the least amount of distortion from the graph.

   Answer:

   The plot should look something like Fig. 8.28.

   If there are M reconstruction levels, then the optimal step size is $\frac{2b_M^*}{M}$, where b_M^* is the value of b_M at which the SNR is maximum on the graph.

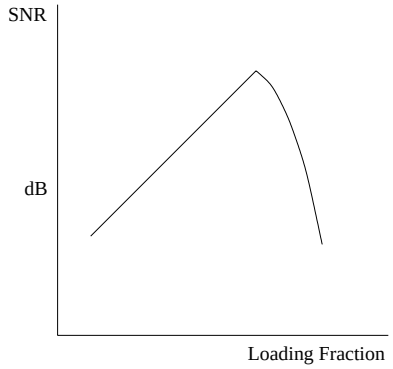


Fig. 8.28: SNR VS Loading Fraction

3. * Suppose the input source is Gaussian-distributed with zero mean and unit variance — that is, the probability density function is defined as

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{8.66}$$

We wish to find a four-level Lloyd–Max quantizer. Let $\mathbf{y}_i = [y_i^0, \ldots, y_i^3]$ and $\mathbf{b}_i = [b_i^0, \ldots, b_i^3]$. The initial reconstruction levels are set to $\mathbf{y}_0 = [-2, -1, 1, 2]$. This source is unbounded, so the outer two boundaries are $+\infty$ and $-\infty$.

Follow the Lloyd–Max algorithm in this chapter: the other boundary values are calculated as the midpoints of the reconstruction values. We now have $\mathbf{b}_0 = [-\infty, -1.5, 0, 1.5, \infty]$. Continue one more iteration for $i = 1$, using Eq. (8.13) and find y_0^1, y_1^1, y_2^1, y_3^1, using numerical integration. Also calculate the squared error of the difference between $\mathbf{y}_1$ and $\mathbf{y}_0$.

Iteration is repeated until the squared error between successive estimates of the reconstruction levels is below some predefined threshold ϵ. Write a small program to implement the Lloyd–Max quantizer described above.

Answer:

The reconstruction values at $i = 1$ is calculated using Equation (8.13). Using numerical integration, we have

$$y_0^1 = \frac{\int_{-\infty}^{-1.5} \frac{x}{\sqrt{2\pi}} e^{-\frac{x}{2}}\, dx}{\int_{-\infty}^{-1.5} \frac{1}{\sqrt{2\pi}} e^{-\frac{x}{2}}\, dx} = -1.94, \quad y_1^1 = \frac{\int_{-1.5}^{0} \frac{x}{\sqrt{2\pi}} e^{-\frac{x}{2}}\, dx}{\int_{-1.5}^{0} \frac{1}{\sqrt{2\pi}} e^{-\frac{x}{2}}\, dx} = -0.62,$$
$$y_2^1 = \frac{\int_{0}^{1.5} \frac{x}{\sqrt{2\pi}} e^{-\frac{x}{2}}\, dx}{\int_{0}^{1.5} \frac{1}{\sqrt{2\pi}} e^{-\frac{x}{2}}\, dx} = 0.62, \qquad y_3^1 = \frac{\int_{1.5}^{\infty} \frac{x}{\sqrt{2\pi}} e^{-\frac{x}{2}}\, dx}{\int_{1.5}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x}{2}}\, dx} = 1.94.$$

The square error of the difference between $\mathbf{y}_1$ and $\mathbf{y}_0$ is computed as

$$(-1.92 + 2)^2 + (-0.62 + 1)^2 + (0.62 - 1)^2 + (1.92 - 1)^2 = 0.296$$

This process is repeated until the square error between successive estimates of the reconstruction levels are below some predefined threshold ϵ.

4. If the block size for a 2D DCT transform is 8×8, and we use only the DC components to create a thumbnail image, what fraction of the original pixels would we be using?

Answer:
1/64, because each 8×8 block only has one DC.

5. When the blocksize is 8, the definition of the DCT is given in Eq. (8.17).

   (a) If an 8×8 grayscale image is in the range $0 \,..\, 255$, what is the largest value a DCT coefficient could be, and for what input image? (Also, state *all* the DCT coefficient values for that image.)

   (b) If we first subtract the value 128 from the whole image and then carry out the DCT, what is the exact effect on the DCT value $F[2, 3]$?

   (c) Why would we carry out that subtraction? Does the subtraction affect the number of bits we need to code the image?

   (d) Would it be possible to invert that subtraction, in the IDCT? If so, how?

   Answer:

   (a) When the image is all WHITE, i.e., all pixels have $I = 255$. The largest coefficient is the DC value which is $8 \times 255 = 2,040$. All others (AC values) are zero.

   (b) There is no effect on $F[2, 3]$. In fact, no effect on any AC values.

   (c) The idea here is to turn it into a zero mean image, so we do not waste any bits in coding the mean value. (Think of an 8×8 block with intensity values ranging from 120 to 135.)

   (d) After decoding, simply add 128 back to all pixel values.

6. We could use a similar DCT scheme for *video streams*, by using a 3D version of DCT.

   Suppose one color-component of a video has pixels f_{ijk} at position (i, j) and time k. How could we define its 3D DCT transform?

   Answer:

$$\begin{aligned} F[u,v,w] &= \frac{1}{4}\sqrt{\frac{2}{N}}C(u)C(v)C(w)\sum_{i=0}^{7}\sum_{j=0}^{7}\sum_{k=0}^{N-1}\cos\frac{(2i+1)u\pi}{16} \\ &\quad \cdot\cos\frac{(2j+1)v\pi}{16}\cdot\cos\frac{(2k+1)w\pi}{2N}\cdot f[i,j,k] \end{aligned}$$

   Must decide on the number N of frames.

7. Suppose a uniformly colored sphere is illuminated and has shading varying smoothly across its surface, as in Fig. 8.29.

   (a) What would you expect the DCT coefficients for its image to look like?

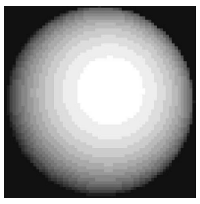


Fig. 8.29: Sphere shaded by a light.

   (b) What would be the effect on the DCT coefficients of having a checkerboard of colors on the surface of the sphere?

(c) For the uniformly colored sphere again, describe the DCT values for a block that straddles the top edge of the sphere, where it meets the black background.

(d) Describe the DCT values for a block that straddles the left edge of the sphere.

Answer:

(a) For the top edge of the sphere, there are DCT values along the left column of each 8×8 DCT block and near zero elsewhere. This is because the cosine functions with $v = 0$ are higher and higher frequencies in the y-direction, and no change in the x-direction; so there is a contribution to these cosines — whereas the cosines with $u = 0$ are sinusoids changing in the x-direction and not in the y-direction, but there is no corresponding change in the x-direction at the top of the sphere, so there result DCTs with values near zero.

For the left (or right) edge of the sphere, there are DCT values along the top row of each 8×8 and near zero elsewhere.

(b) Still lots of low-frequency; but also more higher frequencies because of the sharp edges of the beach ball colors.

(c) For the block that straddle the top edge, as in (a) there are DCT values along the left column of each 8×8 block and near zero elsewhere, although some coefficients in the column, e.g., $F(1, 0)$, will now have much larger (absolute) value in response to the change from black to grey.

(d) For the block that straddle the left edge, this change happens in the top row of each 8×8 block.

8. The Haar wavelet has a scaling function which is defined as follows:

$$\phi(t) \;=\; \begin{cases} 1 & 0 \le t \le 1 \\ 0 & \text{otherwise} \end{cases} \tag{8.67}$$

and its scaling vector is $h_0[0] = h_0[1] = 1/\sqrt{2}$.

(a) Draw the scaling function, and then verify that its dilated translates $\phi(2t)$ and $\phi(2t-1)$ satisfy the dilation equation (8.56). Draw the combination of these functions that make up the full function $\phi(t)$.

(b) Derive the wavelet vector $h_1[0]$, $h_1[1]$ from Eq. (8.59) and then derive and draw the Haar wavelet function $\psi(t)$ from Eq. (8.57).

Answer:

(a) $\phi(t) = \sqrt{2}\,[\frac{1}{\sqrt{2}}\,\phi(2t) + \frac{1}{\sqrt{2}}\,\phi(2t-1)] = \phi(2t) + \phi(2t-1)$, **so that two half-steps make up the full step.**

The two parts are:

$$\phi(2t) \;=\; \begin{cases} 1 & 0 \le t < 0.5 \\ 0 & \textbf{otherwise} \end{cases}$$

and

$$\phi(2t-1) \;=\; \begin{cases} 1 & 0.5 \le t < 1 \\ 0 & \textbf{otherwise} \end{cases}$$

(b) $h_1[0] = h_0[1] = 1/\sqrt{2}, h_1[1] = -h_0[0] = -1/\sqrt{2},$ **so**
$\psi(t) = \sqrt{2}\,[h_1[0]\;\phi(2t) + h_1[1]\;\phi(2t-1)] = \phi(2t) - \phi(2t-1).$

$$\psi(t) \;=\; \begin{cases} 1 & 0 \le t < 0.5 \\ -1 & 0.5 \le t < 1 \\ 0 & \textbf{otherwise} \end{cases}$$

9. Suppose the mother wavelet $\psi(t)$ has vanishing moments M_p up to and including M_n. Expand $f(t)$ in a Taylor series around $t = 0$, up to the nth derivative of f [i.e., up to leftover error of order $O(n+1)$]. Evaluate the summation of integrals produced by substituting the Taylor series into (8.52) and show that the result is of order $O(s^{n+2})$.

   Answer:

   $\mathcal{W}(f,s,u) = \frac{1}{\sqrt{s}} \int f(t)\psi(\frac{t-u}{s})dt$
   $= \frac{1}{\sqrt{s}} \sum_{p=0}^{n} f^{(p)}(0) \int (t^p/p!)\psi((t-u)/s)dt$
   and let $u == 0$**.**

10. The program `wavelet_compression.c` on this book's web site is in fact simple to implement as a MATLAB function (or similar fourth-generation language). The advantage in doing so is that the `imread` function can input image formats of a great many types, and `imwrite` can output as desired. Using the given program as a template, construct a MATLAB program for wavelet-based image reduction, with perhaps the number of wavelet levels being a function parameter.

11. It is interesting to find the Fourier transform of functions, and this is easy if you have available a symbolic manipulation system such as MAPLE. In that language, you can just invoke the `fourier` function and view the answer directly! As an example, try the following code fragment:

```
with('inttrans'); f := 1; F := fourier(f,t,w);
```

    The answer should be $2\pi\delta(w)$. Let's try a Gaussian:

```
f := exp(-t^2);
F := fourier(f,t,w);
```

    Now the answer should be $\sqrt{\pi}e^{(-w^2/4)}$: the Fourier transform of a Gaussian is simply another Gaussian.

12. Suppose we define the wavelet function

$$\psi(t) \;=\; exp(-t^{1/4})\sin(t^4)\;, \quad t \ge 0 \tag{8.68}$$

    This function oscillates about the value 0. Use a plotting package to convince yourself that the function has a zero moment M_p for any value of p.

    Answer:
    To see that function $\psi(t) = e^{(-t^{(1/4)})}\sin(t^4)$ **has all zero moments, use Matlab, or Maple:**

```
% In Matlab: moments:
t = 0:0.001:1000;
t = t';
psi = exp(-t.^(1/4)) .* sin(t.^4);
```

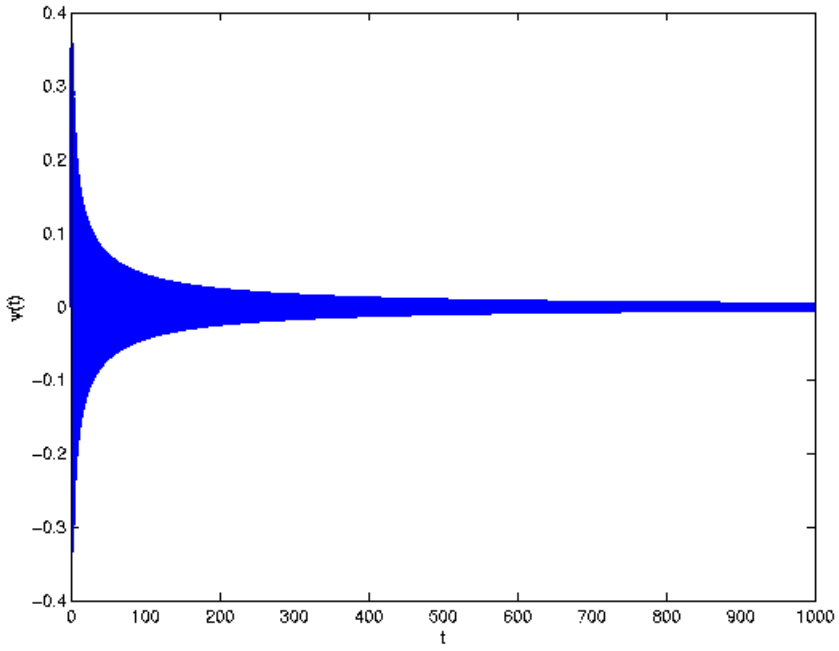


Fig. 8.30: Highly oscillating values function with all zero moments.

plot(t,psi,'.');
xlabel('t');
ylabel('$\psi(t)$');
print -depsc 'zeromoments.eps'; % as in Fig. 8.30.
mom3 = t.^3.*psi;
0.001* sum(mom3)/max(mom3) % approx zero
mom5 = t.^5*psi;
0.001* sum(mom5)/max(mom5) % approx zero

And to see more theoretically,
 # In Maple: moments:
f := exp(-t^(1/4)) * sin(t^4);
#Mp := int(f*t^p, t=0..infinity);
 # Use numerical integration:
M1 := evalf(int(f*t, t=0..1)); # 0.05954452727, and max is 0.3096 in t=0..1
M2 := evalf(int(f*t^2, t=0..1)); # 0.05040557254
M3 := evalf(int(f*t^3, t=0..1)); # 0.04365545560
M7 := evalf(int(f*t^7, t=0..1)); # 0.02829949838

13. Implement both a DCT-based and a wavelet-based image coder. Design your user interface so that the compressed results from both coders can be seen side by side for visual comparison. The PSNR for each coded image should also be shown, for quantitative comparisons.

    Include a slider bar that controls the target bitrate for both coders. As you change the target bitrate, each coder should compress the input image in real time and show the compressed results immediately on your user interface.

Discuss both qualitative and quantitative compression results observed from your program at target bitrates of 4 bpp, 1 bpp, and 0.25 bpp.

Answer:

See example in the Demo part of the web site.

Chapter 9

Image Compression Standards

Exercises

1. (a) JPEG uses the Discrete Cosine Transform (DCT) for image compression.
   i. What is the value of F(0, 0) if the image $f(i, j)$ is as below?
   ii. Which AC coefficient $|F(u, v)|$ is the largest for this $f(i, j)$? Why? Is this $F(u, v)$ positive or negative? Why?

$$\begin{array}{cccccccc}
20 & 20 & 20 & 20 & 20 & 20 & 20 & 20 \\
20 & 20 & 20 & 20 & 20 & 20 & 20 & 20 \\
80 & 80 & 80 & 80 & 80 & 80 & 80 & 80 \\
80 & 80 & 80 & 80 & 80 & 80 & 80 & 80 \\
140 & 140 & 140 & 140 & 140 & 140 & 140 & 140 \\
140 & 140 & 140 & 140 & 140 & 140 & 140 & 140 \\
200 & 200 & 200 & 200 & 200 & 200 & 200 & 200 \\
200 & 200 & 200 & 200 & 200 & 200 & 200 & 200
\end{array}$$

   (b) Show in detail how a three-level hierarchical JPEG will encode the image above, assuming that
   i. The encoder and decoder at all three levels use Lossless JPEG.
   ii. *Reduction* simply averages each 2×2 block into a single pixel value.
   iii. *Expansion* duplicates the single pixel value four times.

Answer:

(a) i. 8 times average-intensity $= 8 \times 110 = 880$**.**

ii. $|F(1, 0)|$ **is the largest, because the intensity value change is similar to a half cosine cycle vertically within the** 8×8 **block.** $F(1, 0)$ **is negative, because the phase of the change is off by 180 degrees. (Or simply put, it is opposite.)**

(b) Step by step results:

X_2**:**

$$\begin{array}{cccc}
\mathbf{20} & \mathbf{20} & \mathbf{20} & \mathbf{20} \\
\mathbf{80} & \mathbf{80} & \mathbf{80} & \mathbf{80} \\
\mathbf{140} & \mathbf{140} & \mathbf{140} & \mathbf{140} \\
\mathbf{200} & \mathbf{200} & \mathbf{200} & \mathbf{200}
\end{array}$$

X_4**:**

$$\begin{array}{cc}
\mathbf{50} & \mathbf{50} \\
\mathbf{170} & \mathbf{170}
\end{array}$$

$E(X_4)$**:**

50	**50**	**50**	**50**
50	**50**	**50**	**50**
170	**170**	**170**	**170**
170	**170**	**170**	**170**

D_2**:**

-30	**-30**	**-30**	**-30**
30	**30**	**30**	**30**
-30	**-30**	**-30**	**-30**
30	**30**	**30**	**30**

$E(X_2) = X_1$
(same as X_1, not shown)

D_2**:**

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Assuming P1 mode of Lossless JPEG (i.e., take the immediate preceding pixel as the predicted value), then the codewords generated are:

X_4**: 50 0 120 0**

D_2**: -30 0 0 0 60 0 0 0 -60 0 0 0 60 0 0 0**

D_1**: 0 0 0 ... 0 0**

2. In JPEG, the Discrete Cosine Ttransform is applied to 8×8 blocks in an image. For now, let's call it DCT-8. Generally, we can define a DCT-N to be applied to $N \times N$ blocks in an image. DCT-N is defined as:

$$F_N(u,v) = \frac{2C(u)C(v)}{N} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N} \cos \frac{(2j+1)v\pi}{2N} f(i,j)$$

$$C(\xi) = \begin{cases} \frac{\sqrt{2}}{2} & \text{for } \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

Given $f(i,j)$ as below, show your work for deriving all pixel values of $F_2(u,v)$. (That is, show the result of applying DCT-2 to the image below.)

$$\begin{array}{cccccccc} 100 & -100 & 100 & -100 & 100 & -100 & 100 & -100 \\ 100 & -100 & 100 & -100 & 100 & -100 & 100 & -100 \\ 100 & -100 & 100 & -100 & 100 & -100 & 100 & -100 \\ 100 & -100 & 100 & -100 & 100 & -100 & 100 & -100 \\ 100 & -100 & 100 & -100 & 100 & -100 & 100 & -100 \\ 100 & -100 & 100 & -100 & 100 & -100 & 100 & -100 \\ 100 & -100 & 100 & -100 & 100 & -100 & 100 & -100 \\ 100 & -100 & 100 & -100 & 100 & -100 & 100 & -100 \end{array}$$

Answer:

Divide the image into 2 by 2 blocks. We only need to work out the four coefficients for $F_2(u, v)$, then they'll repeat.

$F_2(0, 0) = 0$**, because average intensity is zero.**

$F_2(1, 0) = 0$**, because no change vertically.**

$F_2(0, 1) = \frac{\sqrt{2}}{2}\left[\cos\frac{\pi}{4}\cdot 100 + \cos\frac{3\pi}{4}\cdot(-100) + \cos\frac{\pi}{4}\cdot 100 + \cos\frac{3\pi}{4}\cdot(-100)\right] = \frac{\sqrt{2}}{2}\cdot\frac{\sqrt{2}}{2}\cdot 4\cdot 100 = 200.$

$F_2(1, 1) = 0$**, because the signal and the basis function are orthogonal. (Students may check by calculating numbers.)**

$F_2(u, v)$**:**

0	**200**	**0**	**200**	**0**	**200**	**0**	**200**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**200**	**0**	**200**	**0**	**200**	**0**	**200**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**200**	**0**	**200**	**0**	**200**	**0**	**200**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**200**	**0**	**200**	**0**	**200**	**0**	**200**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

3. According to the DCT-N definition above, $F_N(1)$ and $F_N(N-1)$ are the AC coefficients representing the lowest and highest spatial frequencies, respectively.

   (a) It is known that $F_{16}(1)$ and $F_8(1)$ *do not* capture the same (lowest) frequency response in image filtering. Explain why.

   (b) Do $F_{16}(15)$ and $F_8(7)$ capture the same (highest) frequency response?

Answer:

First, we need a 1D DCT-N definition:

$$F_N(u) = \sqrt{\frac{2}{N}}\, C(u) \sum_{i=0}^{N-1} \cos\frac{(2i+1)\cdot u\pi}{2N}\cdot f(i)$$

(a) The basis function for $F_8(1)$ is $\cos\frac{(2i+1)\cdot\pi}{16}$, for $F_{16}(1)$ it is $\cos\frac{(2i+1)\cdot\pi}{32}$. Hence, $F_8(1)$ corresponds to changes of half a cosine cycle within a distance of 16 pixels; whereas $F_{16}(1)$ corresponds to half a cosine cycle within a distance of 32 pixels — a frequency twice as low.

(b) Yes, $F_{16}(15)$ and $F_8(7)$ capture the same (highest) frequency response, because they both capture the highest possible frequency — the 1D signal oscillates between white and black for every pixel, which corresponds to half a cosine cycle within a distance of 1 pixel.

4. You are given a computer cartoon picture and a photograph. If you have a choice of using either JPEG compression or GIF, which compression would you apply for these two images? Justify your answer.

   Answer:

   GIF will generally be better, compared to JPEG, because JPEG firstly keeps 24-bit color, which is not needed for a cartoon, which has only a few numbers, and as well the LZW compression will do a very good job in terms of compression ratio, since only a few values exist and their runs will build up in the dictionary. On the other hand, JPEG will treat the image as a complex image no matter if it is or not.

5. Suppose we view a decompressed 512×512 JPEG image but use only the *color* part of the stored image information, not the luminance part, to decompress. What does the 512×512 color image look like? Assume JPEG is compressed using a 4:2:0 scheme.

   Answer:

   Without all components, we cannot restore the color image. Assuming that we are only recovering the monochrome part, then first, each pixel is an enlarged (duplicated) version of a *subsampled* 2×2 **block, so the image is "pixellated" looking — blocky. Second, although the luminance and chrominance images are often correlated, there is no guarantee that they will capture the same shape and texture information. Depending on the image content, the image displayed may be highly distorted in terms of shape and texture.**

6. (a) How many principal modes does JPEG have? What are their names?
   (b) In the hierarchical model, explain briefly why we must include an encode/decode cycle on the coder side before transmitting difference images to the decode side.
   (c) What are the two methods used to decode only part of the information in a JPEG file, so that the image can be coarsely displayed quickly and iteratively increased in quality?

   Answer:

   (a) 4: baseline sequential, progressive, hierarchical, lossless.
   (b) Because the encoder must only use quantized values that the decoder (receiver) has to operate on.
   (c) 1. Spectral selection – decode using few and then progressively more (and higher-frequency) coefficients of the DCT matrix for each block.
   2. Successive approximation – decode using only the MSB in the DCT representation first, and then progressively add more bits to decode.

7. Could we make use of wavelet-based compression in ordinary JPEG? How?

   Answer:

   Use a series of combinations of Haar-type scaled and shifted wavelets instead of DCT curves which look much like the actual DCT basis functions.

8. We decide to create a new image-compression standard based on JPEG, for use with images that will be viewed by an alien species. What part of the JPEG workflow would we likely have to change?

   Answer:

   The quantization tables, which are based on the human visual system's perceptual importance weighting for different spatial frequencies.

9. Unlike EZW, EBCOT does not explicitly take advantage of the spatial relationships of wavelet coefficients. Instead, it uses the PCRD optimization approach. Discuss the rationale behind this approach.

   Answer:

   By not explicitly exploiting spatial relationships in subbands, the Post Compression Rate Distortion (PCRD) optimization approach is able to offer several advantages. First, the PCRD approach allows subbands to be divided into blocks that are compressed independently. The rate distortion optimization is then performed after all code blocks have been compressed. This gives the encoder a more global view of the code block samples and thus providing better rate distortion tradeoffs.

   Second, the layered coding plus PCRD allows the final bitstream to be both resolution and SNR scalable, whereas algorithms such as EZW can only construct SNR scalable bitstreams.

10. Is the JPEG2000 bitstream SNR scalable? If so, explain how it is achieved using the EBCOT algorithm.

    Answer:

    Depending on the number of quality layers, the JPEG2000 bitstream can be either SNR scalable or not. It is not SNR scalable if there is only one single quality layer. With more than one quality layers, it is SNR scalable. SNR scalability in JPEG2000 is achieved using a combination of layered coding and post compression rate distortion (PCRD) optimization. At a particular quality layers, the PCRD algorithm finds the set of truncation points associated with each code block that contributes to this quality layers. Each additional layer in the bitstream refines the previous layer with the specified truncation points. Thus, the final bitstream is SNR scalable.

11. Implement transform coding, quantization, and hierarchical coding for the encoder and decoder of a three-level Hierarchical JPEG. Your code should include a (minimal) graphical user interface for the purpose of demonstrating your results. You do not need to implement the entropy (lossless) coding part; optionally, you may include any publicly available code for it.

    Answer:

    See some examples in the Demo part of the web site.

Chapter 10

Basic Video Compression Techniques

Exercises

1. Thinking about my large collection of JPEG images (of my family taken in various locales), I decide to unify them and make them more accessible by simply combining them into a big H.261-compressed file. My reasoning is that I can simply use a viewer to step through the file, making a cohesive whole out of my collection. Comment on the utility of this idea, in terms of the compression ratio achievable for the set of images.

   Answer:
   This will not achieve a good compression, since no temporal redundancy is available. And it may be worse, since extra header information is required.

2. In block-based video coding, what takes more effort: compression or decompression? Briefly explain why.

   Answer:
   Compression. The encoder needs to do Motion Compensation (generate the motion vectors) which is time-consuming.

3. An H.261 video has the three color channels Y, C_r, C_b. Should $\mathbf{MV}$s be computed for each channel and then transmitted? Justify your answer. If not, which channel should be used for motion compensation?

   Answer:
   No. MVs are usually generated from the Y-frames. In almost all cases, the luminance (Y) channel carries sufficient motion information.

4. Work out the following problem of 2D Logarithmic Search for motion vectors in detail (see Fig. 10.14).

   The target (current) frame is a P-frame. The size of macroblocks is 4×4. The motion vector is $\mathbf{MV}(\Delta x, \Delta y)$, in which $\Delta x \in [-p, p]$, $\Delta y \in [-p, p]$. In this question, assume $p \equiv 5$.

   The macroblock in question (darkened) in the frame has its upper left corner at (x_t, y_t). It contains 9 dark pixels, each with intensity value 10; the other 7 pixels are part of the background, which has a uniform intensity value of 100. The reference (previous) frame has 8 dark pixels.

   (a) What is the best Δx, Δy, and Mean Absolute Error (MAE) for this macroblock?

   (b) Show step by step how the 2D Logarithmic Search is performed, include the locations and passes of the search and all intermediate Δx, Δy, and MAEs.

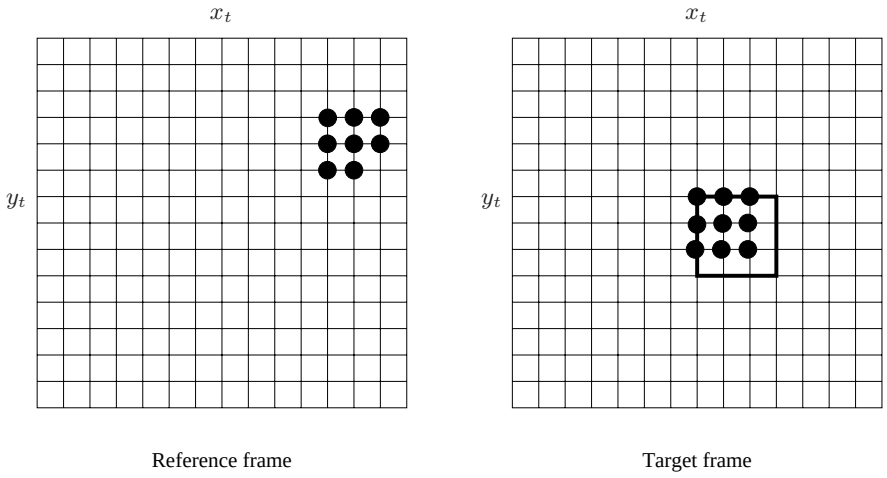


Fig. 10.14: 2D Logarithmic Search for motion vectors.

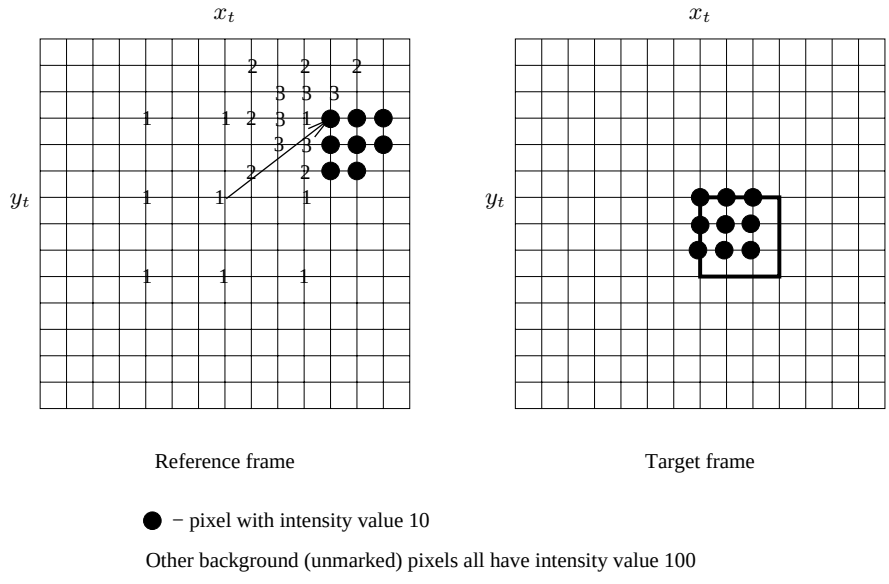
Answer:

(a) $\Delta x = 4, \Delta y = 3$. **MAE =** $\frac{100-10}{16} = 5.625$.

(b) **Pass 1:** $\lceil p/2 \rceil = 3, \Delta x = 3, \Delta y = 3$, **MAE =** $\frac{3 \cdot |100-10| + 2 \cdot |10-100|}{16} = \frac{450}{16} = 28.125$.

Pass 2: $\lceil p/4 \rceil = 2, \Delta x = 3, \Delta y = 3$, **MAE =** $\frac{3 \cdot |100-10| + 2 \cdot |10-100|}{16} = \frac{450}{16} = 28.125$.

Pass 3: $\lceil p/8 \rceil = 1, \Delta x = 4, \Delta y = 3$, **MAE =** $\frac{100-10}{16} = 5.625$.



Answer to Q4: 2D Logarithmic Search for motion vectors.

5. The logarithmic **MV** search method is suboptimal, in that it relies on continuity in the residual frame.

   (a) Explain why that assumption is necessary, and offer a justification for it.

   (b) Give an example where this assumption fails.

(c) Does the hierarchical search method suffer from suboptimality too?

Answer:

(a) The continuity assumption is based on the observation that visual content of the macroblock and the surrounding macroblocks usually change continuously within a short distance, e.g., a couple dozen pixels away, especially within a short time, e.g., 33 milliseconds for 30 fps. This often turns into monotonicity, i.e., the neighborhood that yielded minimal error in the previous pass will indeed yield another (global) minimum in the current pass and beyond.

(b) The above assumption is too strong to be true all the time. In a highly textured frame, it can easily be the case that the (global) minimum actually is in the neighborhood that did not yield minimal error in the previous pass.

(c) Yes, but less so. The textured frame will again be a possible example. When image resolution changes, certain texture patterns change (or disappear). Hence, the area that yielded minimal error at a lower resolution may not necessarily be a good area to search at the current resolution.

6. A video sequence is given to be encoded using H.263 in PB-mode, having a frame size of 4CIF, frame rate of 30 fps, and video length of 90 minutes. The following is known about the compression parameters: on average, two I-frames are encoded per second. The video at the required quality has an I-frame average compression ratio of 10:1, an average P-frame compression ratio twice as good as I-frame, and an average B-frame compression ratio twice as good as P-frame. Assuming the compression parameters include all necessary headers, calculate the encoded video size.

Answer:

Because of the PB-mode, we can assume P- and B-frames always come in pair. Hence, out of 30 frames per second, we have 2 I-frames, 14 P-frames, and 14 B-frames.

4CIF has a resolution of 704×576 **for luminance and** 352×288 **for chrominance images. Assuming 8-bit images, each uncompressed frame has**

$$704 \times 576 + 2(352 \times 288) = 608,256\ bytes \approx 600KB.$$

Consider average compression ratios: I-frame 1/10, P-frame 1/20, B-frame 1/40. For each second, the compressed video has

$$2 \times \frac{1}{10} \times 600 + 14 \times \frac{1}{20} \times 600 + 14 \times \frac{1}{40} \times 600 \approx 750KB.$$

The video size is hence

$$750KB \times 60 \times 90 \approx 4.05GB.$$

7. Assuming a search window of size $2p + 1$, what is the complexity of motion estimation for a QCIF video in the advanced prediction mode of H.263, using

(a) The brute-force (sequential search) method?

(b) The 2D logarithmic method?

(c) The hierarchical method?

Answer:
It is similar to H.261 except the macroblock size N is 8 which doesn't affect the complexity. The QCIF luminance image size is $C \times R = 176 \times 144$. If frame rate is F, then the complexity for each method is as below.

Sequential search method:

$(2p+1)^2 \cdot N^2 \cdot 3 \cdot \frac{C \cdot R}{N \cdot N} \cdot F$, **i.e.,** $O(p^2 \cdot CRF)$.

2D logarithmic method:

$(8 \cdot (\lceil \log_2 p \rceil + 1) + 1) \cdot N^2 \cdot 3 \cdot \frac{C \times R}{N \cdot N} \cdot F$, **i.e.,** $O(\mathbf{log}p \cdot CRF)$.

Hierarchical Search:

$\left[\left(2\left\lceil \frac{p}{4} \right\rceil + 1\right)^2 \left(\frac{N}{4}\right)^2 + 9\left(\frac{N}{2}\right)^2 + 9N^2\right] \times 3 \times \frac{C \times R}{N \cdot N} \times F$

$\left[\left(2\left\lceil \frac{p}{4} \right\rceil + 1\right)^2 \left(\frac{1}{4}\right)^2 + 9\left(\frac{1}{2}\right)^2 + 9\right] \times 3 \times CRF.$ **For a relatively small p (e.g., $p = 15$), this cost is very low.**

8. Discuss how the advanced prediction mode in H.263 achieves better compression.

   Answer:
   As discussed in the text, the macroblock size is reduced from 16×16 to 8×8 and four MVs are generated. A weighted sum of three values is used for any predicted luminance pixel value. This has the potential of generating a more accurate prediction and hence smaller prediction error.

   This trend has continued — in H.264, the macroblock sizes can be even smaller, e.g., 8×4, 4×8, or 4×4.

9. In H.263 motion estimation, the *median* of the motion vectors from three preceding macroblocks (see Fig. 10.11(a)) is used as a prediction for the current macroblock. It can be argued that the median may not necessarily reflect the best prediction. Describe some possible improvements on the current method.

   Answer:
   Do more analysis: e.g., use the MV of the macroblock that has similar color and/or texture as the current macroblock; consistency with other MVs within the VOP, ...

10. H.263+ allows independent forward MVs for B-frames in a PB-frame. Compared to H.263 in PB-mode, what are the tradeoffs? What is the point in having PB joint coding if B-frames have independent motion vectors?

    Answer:
    As said in the text, H.263+ has independent forward motion vectors for B-Frames, just like MPEG, as opposed to H.263 which has P and B frames always together sharing the same motion vectors (joint coding). In PB joint coding you save bits on coding MVs assuming motion is consistent across 2 frames, sacrificing quality if not (thus needing higher bit rate for error coding); while in independent B-frame coding you lower prediction error if motion is jerkey yet you always have to code an additional set of MVs. The main point is to try to do better (MPEG-like) coding yet keeping compatibility with legacy concepts.

Chapter 11

MPEG Video Coding I — MPEG-1 and 2

Exercises

1. As we know, MPEG video compression uses I-, P-, and B-frames. However, the earlier H.261 standard does not use B-frames. Describe a situation in which video compression would not be as effective without B-frames. (Your answer should be different from the one in Fig. 11.1.)

   Answer:

   Besides occlusion, the following could also call for bi-directional search: lighting (color and/or intensity) changes, changing views of 3D shape and/or texture, etc.

2. Suggest an explanation for the reason the default quantization table Q_2 for inter-frames is all constant, as opposed to the default quantization table Q_1 of intra-frames.

   Answer:

   Intra-frames (I-frames) are basically coded as images. As in JPEG, larger values are used in the lower right part of Q_1. As a result, a better compression can be achieved by introducing more losses to higher spatial frequency components.

   For Inter-frames, it is the difference image (not the video frame itself) that is transform coded and quantized. So the reason for using non-constants in Q_2 is not as compelling. Indeed, it can be argued that the high-frequency changes in the difference image are as important as the low-frequency ones.

3. What are some of the enhancements of MPEG-2, compared with MPEG-1? Why hasn't the MPEG-2 standard superseded the MPEG-1 standard?

   Answer:

   MPEG-2 supports higher quality video, usually at a bit-rate of more than 4 Mbps. It introduces much better scalabilities (SNR, Sptial, Temporal and their combination) to support the need of many profiles and levels including DVD and HDTV.

   MPEG-1 was the standard for videos up to 1.5 Mbps. It was aimed at earlier PC and networked video and CD-ROM applications. Indeed, it is gradually replaced by MPEG-4 which also covers low bit-rate applications.

4. B-frames provide obvious coding advantages, such as increase in SNR at low bitrates and bandwidth savings. What are some of the disadvantages of B-frames?

   Answer:

   The obvious one is the overhead — especially the time it needs to do bi-directional search at the encoder side.
   Also, it is hard to make a decision as how to use the results from both the forward and backward predictions: when there are fast motions/transitions, it can often be the case that only one of the motion vectors is accurate. Therefore, a simple average of the difference MBs may not yield good results.

5. The MPEG-1 standard introduced B-frames, and the motion-vector search range has accordingly been increased from $[-15, 15]$ in H.261 to $[-512, 511.5]$. Why was this necessary? Calculate the number of B-frames between consecutive P-frames that would justify this increase.

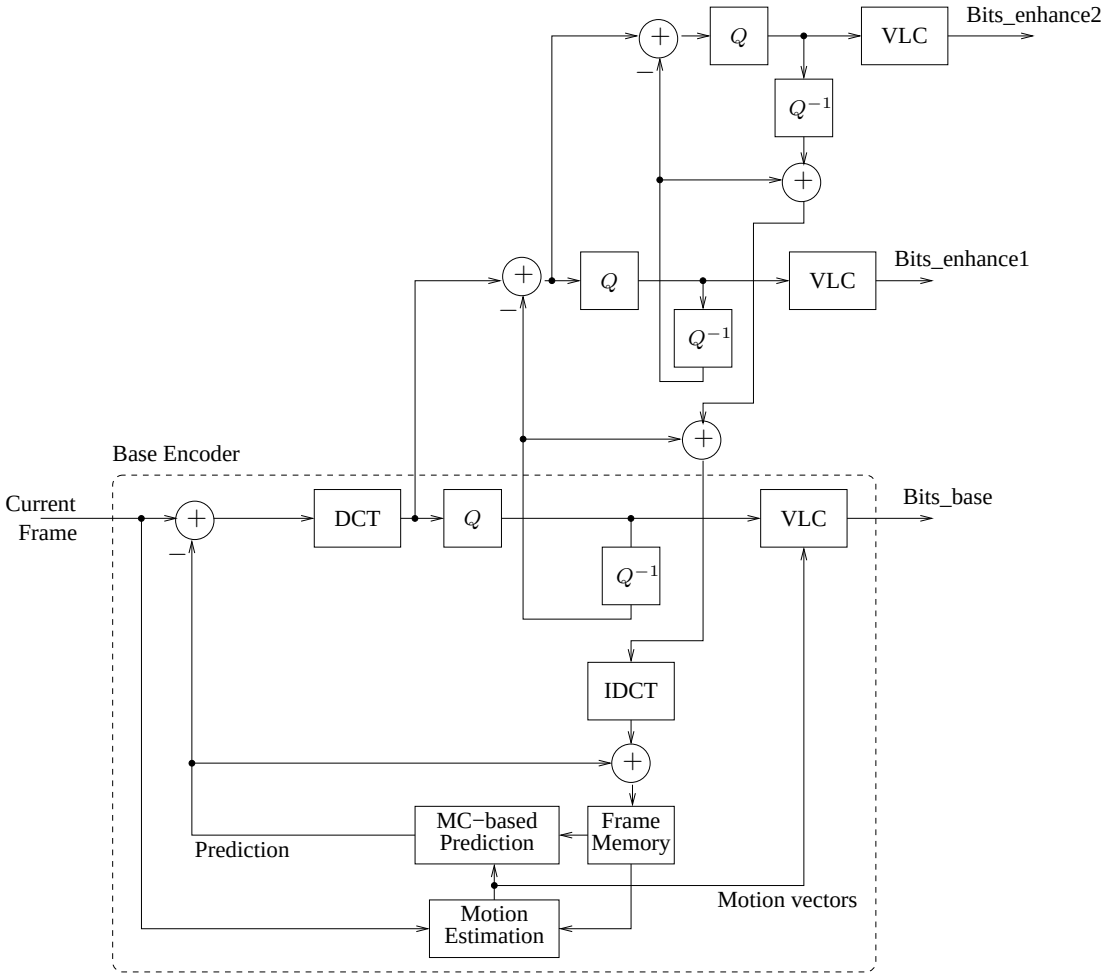
   Answer:

   The range of $[-512, 511.5]$ is used for half-pixel precision. For full-pixel precision it is actually specified as $[-1, 024, 1, 023]$. Both are upper bounds, and may never be used. Note, the larger search window is not only due to the introduction of B-frames. In fact, it is partly made necessary because higher spatial resolution is now supported in MPEG-1 video frames.
   If we simply assume that B-frame is the only cause, then the calculation would suggest that up to $512/15 \approx 34$ B-frames could be in-between consecutive P-frames.

6. Redraw Fig. 11.8 of the MPEG-2 two-layer SNR scalability encoder and decoder to include a second enhancement layer.

   Answer:

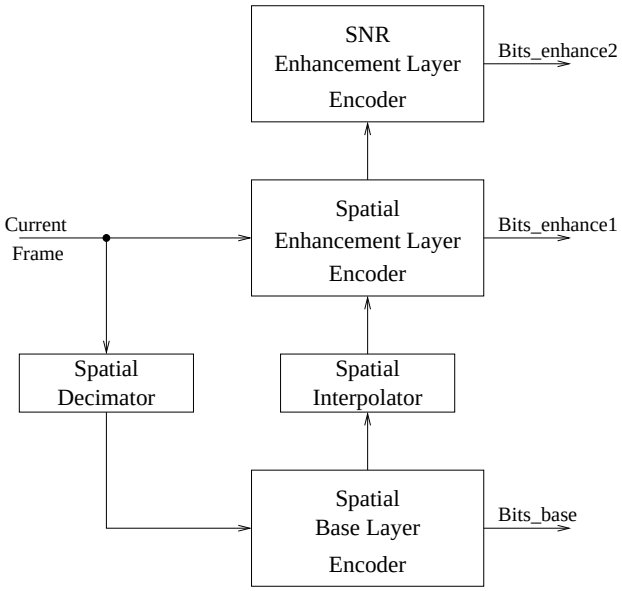
   SNR scalability is based on different levels of quantization on the DCT coefficients of the difference blocks. The following is a three-level diagram.


Bits_enhance2
Q
VLC
Q^{-1}
Bits_enhance1
Q
VLC
Q^{-1}
Base Encoder
Current Frame
DCT
Q
VLC
Bits_base
Q^{-1}
IDCT
MC−based Prediction
Frame Memory
Prediction
Motion vectors
Motion Estimation

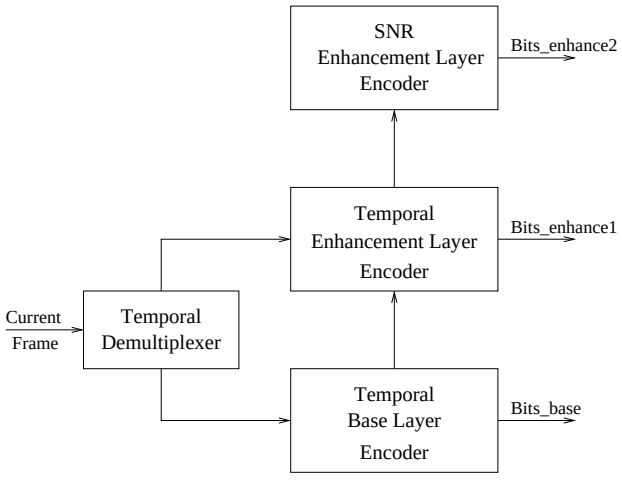
7. Draw block diagrams for an MPEG-2 encoder and decoder for (a) SNR and spatial hybrid scalability, (b) SNR and temporal hybrid scalability.

   Answer:

   (a) Block diagram for SNR and Spatial Hybrid Scalability:



   (b) Block diagram for SNR and Temporal Hybrid Scalability:



8. Why aren't B-frames used as reference frames for motion compensation? Suppose there is a mode where any frame type can be specified as a reference frame. Discuss the tradeoffs of using reference B-frames instead of P-frames in a video sequence (i.e., eliminating P-frames completely).

   Answer:

   As is, the B-frame is at the lowest level of the I-P-B hierarchy. Any error made on P-frame motion compensation (MC) will be carried over to B-frame MC. Therefore, B-frames are not

used as reference frames.

In principle, this can be done (and in fact it is done in later standards.)

It depends on the quality of MC on B-frames. If, for example, B-frames are directly compared to preceding and succeeding I-frames, then the quality will be ensured. The only problem is that we need to introduce more I-frames in order to reduce the distance between B- and I-frames (hence to avoid too large a MV search range).

9. Suggest a method for using motion compensation in the enhancement layer of an SNR-scalable MPEG-2 bitstream. Why isn't that recommended in the MPEG-2 standard?

   Answer:

   A naive method for implementing motion compensation within the enhancement layer(s) of an SNR scalable stream is to treat the enhancement layer as an independent layer (i.e. the enhancement DCT coefficients will only be used for picture reconstruction) and perform motion compensation on it. Note that SNR enhancement layers also have I, P, and B frames.

   This is not a recommended procedure since SNR enhancements are the details of the DCT image, and as such are sensitive to noise and even small contrast changes. It is likely that SNR enhancement layer's consecutive images are not very correlated, and encoding motion vectors for them will cost more than the savings of DCT coding with the motion prediction.

10. Write a program to implement the SNR scalability in MPEG-2. Your program should be able to work on any macroblock using any quantization *step_sizes* and should output both `Bits_base` and `Bits_enhance` bitstreams. The variable-length coding step can be omitted.

    Answer:
    This can be used as a Programming Assignment.

    Sample solution:

```
// MB is an object that can be accessed like array, stores 2D
// integers.
// RefImg is a reference frame for prediction (some image class
// structure, RefImg1 forward, RefImg2 backwards). void
SNRScalability(MB * pMB, int Q1step_size, int Q2step_size) {
  MB DCTMB, Q1DCTMB, Q2DCTMB, IQDCTMB, QMC;

  // Check for frame type, see if temporal prediction necessary.
  if((pMB->frameType == BFRAME) || (pMB->frameType == PFRAME)
  {
    // Calculate the prediction block (in QMB) from the reference
    //frames.
    MotionCompensate(QMB, pMB->MV, pMB->x, pMB->y,
        pMB->predictionType, RefImg1, RefImg2);
    *pMB -= QMB;
  }

  // DCT transform the Macroblock, put output in DCTMB.
  DCT(pMB, DCTMB);
```

```
  // Quantize the DCT base stream, and enhancement stream.
  for(int i=0; i<DCTMB.height; i++)
    for(int j=0; j<DCTMB.width; j++)
      Q1DCTMB[i][j] = DCTMB[i][j] * 8 / Q1step_size;
      Q2DCTMB[i][j] = (DCTMB[i][j] -
         Q1DCTMB[i][j] / 8 * Q1step_size) * 8 / Q2step_size;

  // output both bit streams at this point.
  Q1DCTMB.outputStream();
  Q2DCTMB.outputStream();

  // Calculate the prediction frame.
  IQDCTMB = Q1DCTMB/8*Q1step_size + Q2DCTMB/8*Q2step_size;
  IDCT(&IQDCTMB, QMB);

  // Check for image type to know how to update reference frame.
  if((pMB->frameType == IFRAME) || (pMB->frameType == PFRAME))
  {
    // Check for initial settings.
    if(RefImg1.NotInitialized())
    AddMBtoIMG(RefImg1, QMB);
    // We are assuming here that at when the frame changes
    // (to IFRAME or PFRAME)
    // the encoder copies RefImg2 to RefImg1.
    else
    AddMBtoIMG(RefImg2, QMB);
  }
}
```

Chapter 12

MPEG Video Coding II — MPEG-4, 7 and Beyond

Exercises

1. MPEG-4 motion compensation is supposed to be VOP-based. At the end, the VOP is still divided into macroblocks (interior macroblock, boundary macroblock, etc.) for motion compensation.

   (a) What are the potential problems of the current implementation? How can they be improved?
   Answer:

   - motion vectors for various MBs with the same VOP may not be the same, sometimes they may not even be close.
   + easy, fast
   Possible improvement: Look around each MB in the VOP for continuity. Color, texture, shape (boundary) information can be used to assist in this process, i.e., using multiple cues.

   (b) Can there be true VOP-based motion compensation? How would it compare to the current implementation?
   Answer:

   This is the (3D) object-motion problem that computer vision researcher have been working on for more than three decades now.
   + motion vector closer to object motion, good for indexing, retrieval.
   - slower, could be wrong. The key is object segmentation which is error-prone.

2. MPEG-1 and 2, and 4 are all known as decoder standards. The compression algorithms, hence the details of the encoder, are left open for future improvement and development. For MPEG-4, the major issue of *video object segmentation*, i.e., how to obtain the VOPs, is left unspecified.

   (a) Propose some of your own approaches to video object segmentation.
   Answer:

   Traditional image segmentation methods rely on region growing (based on homogeneous color, texture, etc.), edge/contour detection, and the combination of both. They are known to be unreliable especially in presence of noise, shadow, lighting change.

Digital video (motion picture) has the added temporal (time) dimension. Temporal redundancy (similarity) can be explored to enhance the quality of object segmentation.

Since MPEG video comes with motion vectors for MBs, they can be used to aid object segmentation. Basically, MBs with similar motion vectors can be merged into larger segments. Color, texture, shape (boundary) information can be used to prevent any excessive merge. In other words, an iterative merge-and-split algorithm can be developed.

(b) What are the potential problems of your approach?

Answer:

Object is a difficult thing to recover. Certain features are inevitably lost or distorted in the process of video capturing. It is also not apparent how to incorporate high-level knowledge of objects, such as the presence and behavior of certain objects in certain scenes.

3. Why was padding introduced in MPEG-4 VOP-based coding? Name some potential problems of padding.

Answer:

Padding is introduced to generate pixel values outside an arbitrarily shaped VOP. As stated in the text, (a) Only pixels within the VOP of the current (Target) VOP are considered for matching in motion estimation, and (b) padding only takes place in the Reference VOPs.

This is a quick and cheap way, introduced to gain speed. Some kind of extrapolation method will probably do better.

Since the current padding method makes extensive use of boundary pixel values, if the boundary and the interior of the object are very different or the boundary values are noisy, large matching errors would be introduced.

4. Motion vectors can have subpixel precision. In particular, MPEG-4 allows quarter-pixel precision in the luminance VOPs. Describe an algorithm that will realize this precision.

Answer:

Basically, the same bilinear interpolation method that generated the half-pixel image in Fig. 10.12 can be used one more time to generate quarter-pixel images. Namely, after obtaining half-pixels a, b, c, d as in Fig. 10.12, we can get the quarter-pixels a' = a, b' = (a+b+1)/2, ... This will be followed by the step of motion vector search, now at quarter-pixel precision.

As always, other interpolation methods and/or a larger neighborhood window can also be utilized.

5. As a programming project, compute the SA-DCT for the following 8×8 block:

$$\begin{matrix} 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 4 & 0 & 8 & 16 & 32 & 16 & 8 & 0 \\ 4 & 0 & 16 & 32 & 64 & 32 & 16 & 0 \\ 0 & 0 & 32 & 64 & 128 & 64 & 32 & 0 \\ 4 & 0 & 0 & 32 & 64 & 32 & 0 & 0 \\ 0 & 16 & 0 & 0 & 32 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

6. What is the computational cost of SA-DCT, compared to ordinary DCT? Assume the video object is a 4×4 square in the middle of an 8×8 block.

   Answer:

   They are at the same order. Since SA-DCT operates on fewer input and output values, it is a bit more efficient.

   Using the implementation of 2D separable DCT, the complexity of calculating each DCT coefficient is $O(N)$, where N is the block size. For all DCT coefficients in the block, the complexity is $O(N^3)$.

   In the example above, ordinary DCT has a complexity of $O(8^3)$, whereas SA-DCT has $O(4^3)$.

7. Affine transforms can be combined to yield a composite affine transform. Prove that the composite transform will have exactly the same form of matrix (with $[0\ 0\ 1]^T$ as the last column) and at most 6 degrees of freedom, specified by the parameters a_{11}, a_{21}, a_{31}, a_{12}, a_{22}, a_{32}.

   Answer:

   This is guaranteed as long as the third column for each transform matrix is $[0, 0, 1]^T$. A general form is:

$$\begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & 0 \\ b_{21} & b_{22} & 0 \\ b_{31} & b_{32} & 1 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & 0 \\ c_{21} & c_{22} & 0 \\ c_{31} & c_{32} & 1 \end{bmatrix},$$

   where $c_{11} = a_{11}b_{11} + a_{12}b_{21}$, and $c_{12} = a_{11}b_{12} + a_{12}b_{22}$, ...

8. Mesh-based motion coding works relatively well for 2D animation and face animation. What are the main problems when it is applied to body animation?

   Answer:

   Body is a 3D object and it is also non-rigid (deformable). Due to body motions, occlusion and deformation happen/change which can cause topology changes of the mesh.

9. How does MPEG-4 perform VOP-based motion compensation? Outline the necessary steps and draw a block diagram illustrating the data flow.

   Answer:

   The details of defining VOP and padding are described in the text. Fig. 12.21 below is more than a block diagram, but is probably needed.

10. What is the major motivation behind the development of MPEG-7? Give three examples of real-world applications that may benefit from MPEG-7.

    Answer:

    Indexing and retrieval of multimedia databases and digital libraries.

    Content-based image and video retrieval (broadcast programs and/or film archiving, museum and/or library paintings and books), e-commerce, tele-learning, tele-medicine, ...

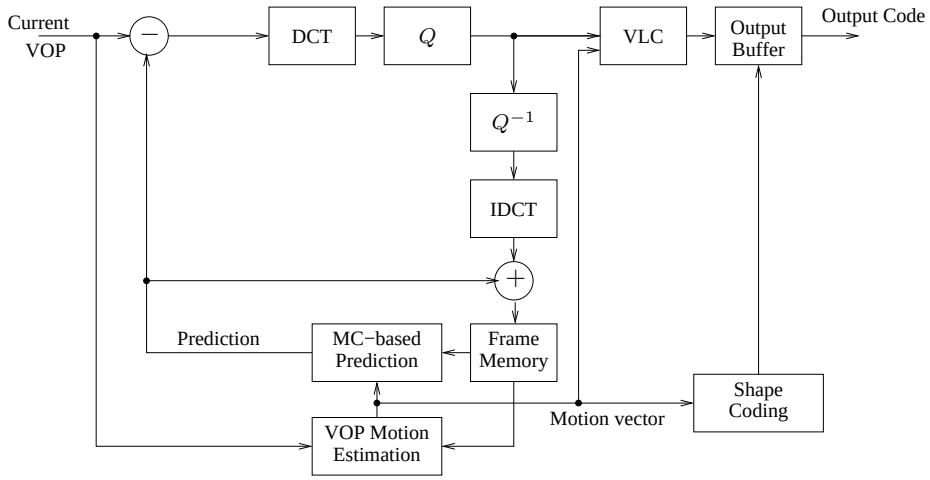


Fig. 12.21 MPEG-4 VOP-based motion compensation.

11. Two of the main shape descriptors in MPEG-7 are "region-based" and "contour-based". There are, of course, numerous ways of describing the shape of regions and contours.

    (a) What would be your favorite shape descriptor?

    (b) How would it compare to ART and CSS in MPEG-7?

    Answer:

    (a) Our favorite is *locale* — the *feature localization* method that we described in Chapter 18.

    (b) CSS is invariant to translations and rotations, and it is pretty robust to scale changes. However, it relies on a good contour extraction which is (almost) non-attainable. ART is a set of descriptors at a pretty high level. It lacks the descriptive power (not quite usable).

    The locale-based descriptor is not based on "image segmentation". Instead, it attempts to localize features. Therefore it has a better chance to survive when a "segmentation" effort fails. When needed, locales can also have their own shape descriptors down to the pixel precision.

Chapter 13

Basic Audio Compression Techniques

Exercises

1. In Section 13.3.1 we discuss phase insensitivity. Explain the meaning of the term "phase" in regard to individual frequency components in a composite signal.

   Answer:
   The relative delay of individual frequency components.

2. Input a speech segment, using C or MATLAB, and verify that formants indeed exist — that any speech segment has only a few important frequencies. Also, verify that formants change as the interval of speech being examined changes.

   A simple approach to coding a frequency analyzer is to reuse the DCT coding ideas we have previously considered in Section 8.5. In one dimension, the DCT transform reads

   $$F(u) = \sqrt{\frac{2}{N}}\, C(u) \sum_{i=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N}\, f(i), \tag{13.35}$$

   where $i, u = 0, 1, \ldots, N-1$, and the constants $C(u)$ are given by

   $$C(u) = \begin{cases} \frac{\sqrt{2}}{2} & if \quad u = 0, \\ 1 & otherwise. \end{cases} \tag{13.36}$$

   If we use the speech sample in Fig. 6.15, then taking the one-dimensional DCT of the first, or last, 40 msec (i.e., 32 samples), we arrive at the absolute frequency components as in Fig. 13.5.

   Answer:

```
% formants.m
% matlab script
load 'resources_exercises/chap13/noahwav.txt' -ascii; % 8000 samples.
% look for formants:
noahwav1 = noahwav(1:32); % beginning 40 msec
noahwav2 = noahwav(7969:8000);  % last 40 msec
formants1 = dct(noahwav1); % blocksize is same as noahwav1, i.e. 32.
formants2 = dct(noahwav2);
%
```

```
plot(abs(formants1));
hold on
plot(abs(formants2),'--');
xlabel('Frequency');
ylabel('abs( Coefficient)');
hold off
```

3. Write code to read a WAV file. You will need the following set of definitions: a WAV file begins with a 44-byte header, in unsigned byte format. Some important parameter information is coded as follows:

Byte[22..23]	Number of channels
Byte[24..27]	Sampling Rate
Byte[34..35]	Sampling Bits
Byte[40..43]	Data Length

   Answer:
   A WAVE file is a RIFF file. The complete WAV format is at
   http://www.wotsit.org/download.asp?f=wavecomp
   The code for WavRead.cpp and WavRead.h can be downloaded from
   http://www.microsoft.com/msdownload/platformsdk/Samples/Multimedia/DSound/Src
   An older, 1999, version from Microsoft is at
   http://140.131.13.205/LHU_EL_Teacher/el049/DirectX/DX7ASDK/DXF/samples/multimedia/dsound/src/voicemanagement/
   and that version is copied into `resources_exercises/chap13/wavread.zip`**.**

4. Write a program to add fade in and fade out effects to sound clips (in WAV format). Specifications for the fades are as follows: The algorithm assumes a linear envelope; the fade-in duration is from 0% to 20% of the data samples; the fade-out duration is from 80% to 100% of the data samples.

   If you like, you can make your code able to handle both mono and stereo WAV files. If necessary, impose a limit on the size of the input file, say 16 megabytes.

   Answer:

```
% wavfade.m
% matlab script
wav = wavread('resources_exercises/chap13/coffee.wav');
wav = double(wav);
ss = size(wav,1); % 16000
fadesize = fix(ss/5);
fadefnc = ( (1:fadesize)/fadesize )' ;
fadefnc2 = (fliplr(fadefnc'))';
wavoutp = wav;
intro = wav(1:fadesize).*fadefnc;
outro = wav((ss-fadesize+1):ss).*fadefnc2;
wavoutp(1:fadesize) = intro;
wavoutp((ss-fadesize+1):ss) = outro;
plot(wav)
```

```
plot(wavoutp)
wavwrite(wavoutp, 'resources_exercises/chap13/coffeefade.wav');
```

5. In the text, we study an adaptive quantization scheme for ADPCM. We can also use an adaptive prediction scheme. We consider the case of one tap prediction, $\hat{s}(n) = a \cdot s(n-1)$. Show how to estimate the parameter a in an open-loop method. Estimate the SNR gain you can get, compared to the direct PCM method based on a uniform quantization scheme.

   Answer:
   To estimate a, we minimize the following function

$$E[e^2] = E[(s(n) - as(n-1))^2]$$

   by $\partial E[e^2]/\partial a = E[(s(n) - as(n-1))(-s(n-1))] = 0$**. The estimate of a is**

$$\begin{aligned} a &= E[s(n)s(n-1)]/E[s^2(n)] \\ &= R(1)/\sigma_s^2 \end{aligned}$$

   In ADPCM, $s(n) = e(n) + \tilde{s}(n) = e_q(n) + e_r(n) + \tilde{s}(n)$**, where $\tilde{s}(n)$ is the prediction signal, $e_q(n)$ is the quantized difference signal of $s(n)$ and $\tilde{s}(n)$ and $e_r(n)$ is the quantization error of $e(n)$. Therefore, the quantization error in ADPCM is equal to the quantization error of $e(n)$. For uniform quantization, the variance of the quantization error is proportional to the variance of the input signal. Thus,**

$$\begin{aligned} E[e_r^2] \simeq E[e^2] &= E[(s(n) - as(n-1))^2] \\ &= \sigma_s^2(1 - a^2) \end{aligned}$$

   where $a = R(1)/\sigma_s^2$**. Usually a is strictly less than 1. Thus the quantization SNR gain is** $-10\log 10(1 - a^2)$**.**

6. Linear prediction analysis can be used to estimate the shape of the envelope of the short-time spectrum. Given ten LP coefficients $a_1, \ldots, a_{10}$, how do we get the formant position and bandwidth?

   Answer:
   Solve the equation $1 + \sum_{i=1}^{10} a_i z^{-i} = 0$ **and obtain the roots** $\{z_i = r_i e^{j\theta_i}, i = 1, 2...10\}$**. Then the normalized frequency of the ith formant is $F_i = \theta_i/(2\pi)$ and its bandwidth is $B_i = \log r_i/\pi$.**

7. Download and implement a CELP coder (see the textbook web site). Try out this speech coder on your own recorded sounds.

8. In quantizing LSP vectors in G.723.1, splitting vector quantization is used: if the dimensionality of LSP is 10, we can split the vector into three subvectors of length 3, 3, and 4 each and use vector quantization for the subvectors separately. Compare the codebook space complexity with and without split vector quantization. Give the codebook searching time complexity improvement by using splitting vector quantization.

   Answer:
   If we use 8 bits for each subvector codebook, we need 24 bits for the LSP vector. The space used is pow(2, 8)* 10 * size(float). With general VQ, the space used is pow(2, 24) * 10 * size(float). The codebook searching time is proportional to the codebook size.

9. Discuss the advantage of using an algebraic codebook in CELP coding.

   Answer:
   Firstly, we do not need training. Second there, are many zeros and overlap between different codewords, so that fast codebook searching is possible.

10. The LPC-10 speech coder's quality deteriorates rapidly with strong background noise. Discuss why MELP works better in the same noisy conditions.

    Answer:
    LPC-10 uses a binary U/V decision, which can give a wrong decision and degrade the synthesized speech. MELP uses a multi-model U/V decision. In fact, it uses a voice degree to describe the ratio of noise and periodic components in each band. This can give a much better fit to real, noisy, input speech.

11. Give a simple time-domain method for pitch estimation based on the autocorrelation function. What problem will this simple scheme have when based on one speech frame? If we have three speech frames, including a previous frame and a future frame, how can we improve the estimation result?

    Answer:
    Find the peaks of the autocorrelation function. If we have multi-frames we can use dynamic programming and delay the decision.

    That is, in the multi-frame situation we can choose a pitch based not only on the autocorrelation function from a single frame but instead based on several frames by adding a smoothness constraint. For example, we can choose 10 local peaks of the autocorrelation function as the pitch candidates in the current frame and in the future frame. Since the pitch in the previous frame is already determined, our task now is to find a solution (three pitches) in the three successive frames which optimizes some cost function. One naive solution is by exhaustively searching $10\times10=100$ possibilities. A better method involves dynamic programming.

    A student solution includes defining some energy function, and there are many possible candidates.

12. On the receiver side, speech is usually generated based on two frames' parameters instead of one, to avoid abrupt transitions. Give two possible methods to obtain smooth transitions. Use the LPC codec to illustrate your idea.

    Answer:
    One method is interpolating the parameters for consecutive frames and forming a time-varying synthesis filter to process the excitation. The second method synthesizes the speech in each frame, separately and overlapping, with addition by some appropriate windowing.

Chapter 14

MPEG Audio Compression

Exercises

1. (a) What is the threshold in quiet, according to Eq. (14.1), at 1,000 Hz? (Recall that this equation uses 2 kHz as the reference for the 0 dB level.)

   Answer:
   The threshold in quiet is:

   $$3.64 * (f/1000)^{-0.8} - 6.5 * exp(-0.6 * (f/1000 - 3.3)^2) + 10^{-3} * (f/1000)^4$$

   At $f = 1000$, this evaluates to 3.369067.

   (b) Take the derivative of Eq. (14.1) and set it equal to zero, to determine the frequency at which the curve is minimum. What frequency are we most sensitive to? Hint: One has to solve this numerically.

   Answer:

```
# Maple script:
tiq := 3.64*f^(-0.8) - 6.5*exp( (-0.6)*(f-3.3)^2 ) + 10^(-3)*f^4;
# f in kHz
df:=diff(tiq,f);
fsolve(df,f);
# 3.324133041
== 3324.133041 Hz
```

2. Loudness versus Amplitude. Which is louder: a 1,000 Hz sound at 60 dB or a 100 Hz sound at 60 dB?

   Answer:
   The 1000 Hz sound at 60 dB will have loudness at a phon level of 60dB.
   The 100 Hz sound will be perceived as a phon level of approximately 50 dB so the 1000 Hz sound will be much louder.

3. For the (newer versions of the) Fletcher-Munson curves, in Fig. 14.1, the way this data is actually observed is by setting the y-axis value, the sound pressure level, and measuring a human's estimation of the effective perceived loudness. Given the set of observations, what must we do to turn these into the set of perceived loudness curves shown in the figure?

Answer:
Invert the functions — just use linear interpolation. In detail, according to the Robinson-Dadson paper (and others), we can capture the received loudness (phons, P) as a function of the stimulus D (also in dB) via a function $P = a + bD + cD^2$, where a,b,c are functions of frequency f. Thus for each f we have a set of values P(D), with D=0:10:120, say. Now we'd like to develop a second set of values, D=D(P), for equally-spaced Phon values P. So we interpolate. The interpolation is unworkable on the very low and very high D ends, unless we use better than linear interpolation.

4. Two tones are played together. Suppose tone 1 is fixed, but tone 2 has a frequency that can vary. The *critical bandwidth* for tone 1 is the frequency range for tone 2 over which we hear *beats*, and a roughness in the sound. Beats are overtones at a lower frequency than the two close tones; they arise from the difference in frequencies of the two tones. The critical bandwidth is bounded by frequencies beyond which the two tones sound with two distinct pitches.

   (a) What would be a rough estimate of the critical bandwidth at 220 Hz?

   Answer:
   According to eq. (14.5), the critical bandwidth (df) for a given center frequency f can also be approximated by

   $$df \;=\; 25 \;+\; 75 \times [1 + 1.4(f^2)]^{0.69} \quad ,$$

   where f is in kHz and df is in Hz.

   According to this formula, at 0.22 kHz, the bandwidth is roughly 103.5 Hz.

   (b) Explain in words how you would set up an experiment to measure the critical bandwidth.

   Answer:
   Dr. David Brainard (in `SoundThreshWriteup.pdf`) writes: "The idea behind critical bands is that sounds at different frequencies are processed by different auditory channels or mechanisms. If this is so, then a masker at one frequency will not mask a test at another frequency that is sufficiently different, since if the two differ in frequency they will be processed by different channels. To test this, you would measure the threshold to detect a tone in the presence of a masker at different frequencies. For this experiment, it is a good idea to set the test bandwidth to zero and the masker bandwidth to about 200 or so. You then measure the threshold T as a function of the masker frequency F. There should be a lot of masking (a high threshold) when masker has the same frequency as the test. But as the frequency of the masker increases or decreases from the test, we expect the threshold for the test to get smaller, producing an inverted U-shaped curve when T is plotted against F. The width of the U-shape curve tells us the range of frequencies that the mechanism detecting the test is sensitive to. This critical bandwidth is thought to increase with test frequency, and a really nice experiment would be to try to measure this increase in bandwidth."

5. Search the web to discover what is meant by the following psychoacoustic phenomena:

   (a) Virtual Pitch

   Answer:
   A pure sine wave has so-called "spectral pitch" that is easy to comprehend. In contrast, "virtual pitch" refers to the basic pitch a himan senses for a complex harmonic tone. In this case, there is not any "actual" specific pitch, but just a general one that is perceived— it is not really there, so is "virtual". An example is male speech, which is perceived as a

bass tone rising and falling, notwithstanding the fact that there is a complex wave form actually present.

A more complex version of this is that if several harmonics are present, but the fundamental is filtered out, one can still perceive the sound as belonging to the fundamental, rather than the harmonics that are in fact present! The sound perceived in this case depends on how the harmonics are spaced out: the wider the spacing between harmonics that are allowed to remain in the signal, the higher is the virtual pitch perceived.

(b) Auditory scene analysis

Answer:

"a process in which the auditory system takes the mixture of sound that it derives from a complex natural environment and sorts it into packages of acoustic evidence in which each package probably has arisen from a single source of sound."

In particular, auditory scene analysis is concerned with computer modeling of the process by which humans convert complex sound into distinct, interpreted abstractions such as the words of a particular speaker. The computational problem often comes down to separating speech from interfering noise.

(c) Octave related complex tones

Answer:

Ascending or descending "staircases" of tones make us perceive an ambiguity in the pitch: these "octave-related complex tones" are in fact perceived by us via by following the pitch that is moving. The ambiguity is to ascertain just what octave to place a sound into, if several harmonics are present at the same time. Our way out of the problem is to simply choose the moving tone, if there is one. Shepard Scales consist of such harmonics, that include a "staircase" – and that is the tone that one automatically hears best.

(d) Tri-tone paradox

Answer:

This phenomenon again has to do with octave related sinewave components. Whereas Shepard tones consist of 10 components (octaves) with magnitudes weighted by a Gaussian over log-frequency, the tones used in the tritone paradox have 6 components (octaves) with weights given by a filter that cuts off more sharply, again over log-frequency.

Here, pairs of tones are played together. When the second tone is less than one half an octave (an interval called the "tritone" — e.g., C to F#) above the first, we hear the second tone as higher in pitch compared to the first (e.g., C to E is less than a tritone). But if the second tone is more than a tritone above the first, (e.g., C to G is more than a tritone) we perceive the second as lower in pitch compared to the first.

(e) Inharmonic complex tones

Answer:

Another paradox. Here we listen to a set of non-harmonics: tones that are separated by some particualr, but not special, frequency difference. First we hear a set of six tones each separated by the same difference, and then the whole set is moved up in frequency, by a small step. At some point,at the top of the frequency range represented, the whole set is re-initialized back where it started and the march upwards recommences. The result: we insist on hearing the higher tones, i.e., we hear the staricase of sound go on ascending even though in fact it has been re-initialized.

6. If the sampling rate f_s is 32 ksps then, in MPEG Audio Layer 1, what is the width in frequency of each of the 32 subbands?

Answer:
500 Hz.

7. Given that the level of a *masking tone* at the 8th band is 60 dB, and 10 msec after it stops, the masking effect to the 9th band is 25 dB.

   (a) What would MP3 do if the original signal at the 9th band is at 40 dB?

   (b) What if the original signal is at 20 dB?

   (c) How many bits should be allocated to the 9th band in (a) and (b) above?

   Answer:
   Only send 40-25 = 15 dB.
   Send 3 bits instead of 7 bits → saving of 4 bits.
   Send no bit.

8. What does MPEG Layer 3 (MP3) audio do differently from Layer 1 to incorporate temporal masking?

   Answer:
   More (1,152) samples per frame. Includes psychoacoustic model with temporal masking effects.

9. Explain MP3 in a few paragraphs, for an audience of consumer-audio-equipment salespeople.

   Answer:
   MPEG audio layer 3 is a type of audio codec for achieving significant compression from the original audio source with very little loss in sound quality. A compression ratio of up to 12:1 produces very little degradation. The standard bit rate ("near-CD" quality) is 128 or 112 kbit/s. An advantage of MP3 is that files can be broken up into pieces, with each piece is still playable. The feature that makes this possible (headerless file format) also means that MP3 files can be streamed in real-time. A disadvantage of MP3 compression is that high processor power is required to encode and play files in software. Hardware player/encoder/decoders are still quite expensive.

10. Implement MDCT, just for a single 36-sample signal, and compare the frequency results to those from DCT. For low-frequency sound, which does better at concentrating the energy in the first few coefficients?

    Answer:
    Solutions for forward and inverse MDCT, using straightforward loops and also using a more vector-based matlab approach, and in `resources_exercises/chap14/` **as** `mdctl.m`**,** `imdctl.m`**,** `mdctv.m`**, and** `imdctv.m`**.**

    Suppose we take a small segment of sound, as follows: Then we find, below, that the DCT does a considerably better job at concentrating the energy for a general signal. However, while not placing energy in a single first coefficient, the MDCT does much better overall in concentrating the energy in the first few coefficients for a *smooth* signal.

```
% matlab script
sig = rand(8); % uniform random, 8x8 matrix
sig = sig(:,1); % 8x1
% 0.5869 0.0576 0.3676 0.6315 0.7176 0.6927 0.0841 0.4544
% sig = [0.5869 0.0576 0.3676 0.6315 0.7176 0.6927 0.0841
%        0.4544]';
```

```
sigd = dct(sig);
% 1.2701 -0.0447 -0.3180 0.2411 0.4202 -0.0177 0.3654 -0.0726
% 1st component large compared to others.
abs(sigd(1))/max(sigd(2:end)) % 3.0229
sigm = mdctv(sig);
% -1.8215 -0.2013 0.4515 0.7666
abs(sigm(1))/max(sigm(2:end)) % 2.3764
%
% and on a bigger set:
%
sig = rand(100); % uniform random, 100x100 matrix
sig = sig(:); % 10,000x1
sigd = dct(sig); % length==10,000
abs(sigd(1))/max(sigd(2:end)) % 46.6140
sigm = mdctv(sig); % length==5,000
abs(sigm(1))/max(sigm(2:end)) % 5.2018
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Low-freq:
% on a small low-freq signal:
%
sig = randn(36); % gaussian random, 36x36 matrix
sig = sig(:,1); % 36x1
sig = sort(sig);
sigd = dct(sig); % length==36
abs(sigd(1))/max(sigd(2:end)) % 3.8880
sigm = mdctv(sig); % length==18
abs(sigm(1))/max(sigm(2:end)) % 4.7657
%
sig = randn(100); % gaussian random, 100x100 matrix
sig = sig(:); % 10,000x1
sig = sort(sig);
plot(sig); % smooth....
sigd = dct(sig); % length==10,000
abs(sigd(1))/max(sigd(2:end)) % 4.9593
abs(sigd(1))/max(sigd(20:end)) % 16.7562
sigm = mdctv(sig); % length==5,000
abs(sigm(1))/max(sigm(2:end)) % 3.9110
abs(sigm(1))/max(sigm(20:end)) % 87.1210
```

11. Convert a CD-audio cut to MP3. Compare the audio quality of the original and the compressed version — can you hear the difference? (Many people cannot.)

    Answer:
    The answer depends on the compression level set. According to http://www.audioboxinc.com/quality.html,
    “A majority of listeners can distinguish the difference between an MP3 audio file, at any compression level, and an uncompressed audio file. A majority of listeners can hear quality loss

from a TAPE RECORDING of an MP3 file at any compression level higher than 10:1."

12. For two stereo channels, we would like to be able to use the fact that the second channel behaves, usually, in a parallel fashion to the first, and apply information gleaned from the first channel to compression of the second. Discuss how you think this might proceed.

    Answer:
    One simple approach is to encode one channel, plus the sum L+R. In terms of MPEG Audio, we would then assign independent left- and right-channel scalefactors.

Chapter 15

Computer and Multimedia Networks

Exercises

1. What is the main difference between the OSI and TCP/IP Reference Models?

   Answer:

   The ISO OSI reference model lists 7 logically separate layers to be implemented for a complete network connectivity. Each layer can be independently implemented by a different source in accordance with any technology, and they must all inter-operate according to the OSI specifications. In contrast, the TCP/IP protocol stack was developed as a practical solution to interfacing separate networks where the particular technology used is clearly defined (hence interoperability is of less concern). It is intended to have defined transport and network layers over all types of existing networks in order to communicate with different network types, however there is one defined transport layer for applications to interface to. The TCP/IP stack has fewer layers, thus easier to implement, and the application layer was developed separately on top of the protocol suite itself.

2. IPv6 is a newer IP protocol. What is its advantage over IPv4?

   Answer:
   IPv6 is a "next generation" protocol designed to replace IPv4. IPv6 supports 128bit IP addresses as opposed to the 32bit address field length in IPv4. In addition IPv6 defines flow control, multicasting, security and allows for extending the usage of the IPv6 headers in the future. Additional flow control and QoS support is achieved using a labelling of packets that have other priorities than default (the labels assist in routing). Multicasting is supported by adding a "scope" field to the IPv6 header, and a new address called "anycast address" is defined to indicate sending a packet to all group nodes.

3. UDP does not provide end-to-end flow control, but TCP does. Explain how this is achieved using sequence numbers. Give an example where a packetized message sent using UDP is received incorrectly, but when using TCP it is received correctly under the same circumstances (without channel errors).

   Answer:
   As the message is packetized by the TCP layer the byte count up to the first byte in the current packet is specified in the sequence number field, and the packet is sent to the receiving station. The communicating station also has a "window" buffer for storing TCP packets until they are

read by the client software; when the buffer is read by software, the window slides forward to have the next TCP packet be the first packet in the buffer. As TCP packets arrive they are not put in the window buffer according to arrival order but rather according to sequence numbers. This can generate holes in the buffer, at least until the correct packet arrives. The "ACK" packet that is sent contains the sequence number that the packet filling the first hole should have. If no holes exist then the sequence number of the next packet that will be received is sent. The "ACK" packet also contains in the window field the amount of additional data the window buffer can hold (after the last packet received).

The sending station will also have packets in its window buffer, and they are removed only after "ACK" for them has been received, hence the station can only send the minimum of the data in its window that has not yet been sent and the window size of the receiving station. If an "ACK" for a packet sequence number has not been received within a timeout period after sending it (or if a negative "ACK" packet received) the packet is transmitted again. In this way flow control and correct packet ordering is achieved.

In a hypothetical scenario where a station sending packets (UDP or TCP) to another station across at least a few routers and one of the routers has updated its routing tables due to congestion to rerout the message through different routers, the first few packets in that rest of message could arrive before the last few packets in the first part of the message (due to congestion). If the packet were encoded using UDP, they would be read out of sequence, however with TCP they are reordered correctly even if earlier packets arrive late.

4. As a variation of FDM, WDM is used for multiplexing over fiber-optic channels. Compare WDM with FDM.

   Answer:
   WDM is really FDM applied to light frequencies. Although light is attributed more properties than other electromagnetic waves, the same exact principles for FDM apply to WDM. While FDM would normally be transmitted in the air using antennas/satellite dishes and coaxial cable on the ground, WDM is transmitted using lasers and optical fibers on the ground. FDM transmission is prone to more scattering and interference and needs electronic devices prone to noise to demultiplex the signal. WDM transmission over fiber has minimal scattering, not prone to self or outside interference, harder to eavesdrop and is handled using optical devices who are analog and less prone to noise. Thus it can achieve much higher bandwidth.

5. Both ISDN and ADSL deliver integrated network services, such as voice, video, and so on, to home users or small-office users. What are the advantages of ADSL over ISDN?

   Answer:
   ADSL has a few advantages over ISDN, it is cheaper, and theoretically can carry ten times as much data as Primary Rate ISDN. The key principle that gives it the edge is however that ADSL is much higher bandwidth downstream than upstream, while in ISDN it is symmetric. It is observed that home users download much more than they need to upload, which lets ISPs have advanced circuits on the server side only. Additionally ADSL uses the DMT technology to determine the proper amount of data to send on each subchannel so that line interference is handled more effectively over twisted pair wires.

6. Several protocols, such as Ethernet, Token ring, and FDDI, are commonly used in LAN. Discuss the functionalities of these three technologies and differences among them.

   Answer:
   The three protocols, Ethernet, Token Ring, and FDDI are used mainly in the data link layer

and are responsible for specifying how computers can communicate with each other over the network. The problem with communications on a network is obviously "cross-talk" when two computers want to send a message at the same time over a shared medium. FDDI is specified over optical fiber lines while Token Ring and Ethernet have been designed for copper wires, such as twisted-pair although newer standards are adapted to fiber.

Ethernet handles access control in a very different way than Token Ring and FDDI which is derived from it. While both Ethernet and Token Ring can physically have the same bus, in Ethernet a station wanting to transmit will listen to the line and if clear will transmit, monitoring for collisions (CSMA/CD). If there is a collision the retransmission will occur after a random delay (with line sensing first) so that the same collision won't repeat itself. In Token Ring there can be no collisions, since only the station with a token is allowed to transmit, and then the token is not released to the next waiting station until the transmission is done. Ethernet thus has the ability to send packets with no delay and achieve a high throughput, however as more stations need to transmit at the same time it can get bogged down with collisions/retransmissions. On the other hand a token ring always behaves orderly but has a larger delay for sending each packet of data, effectively reducing transmission rate. FDDI has a dual ring topology for error tolerance since it is being used over wide distances, and is basically a Token Ring except tokens are released immediately after transmitting as many frames as allowed in the time frame. It also has a synchronous mode with bandwidth reservation. It achieves much higher rates than Token Ring, but still has the problem of latency for transmitting.

7. Frame relay and Cell relay are variants of packet switching. Compare these two technologies.

   Answer:
   In designing Frame Relay the objective was to increase throughput by reducing packet header overhead and extraneous error checks. In order to increase the header bit-rate impact the Frame has to be relatively large in the order of 1KB. Cell relay on the other hand is designed to achieve lower routing delay, and so the cell sizes have to be small, chosen as 53 bytes, with a 5 byte header, which is nearly ten percent overhead. However, cell relay also supports error checking on the header (for correct routing) and some priority routing for QoS support. Both Cell Relay and Frame Relay are designed to work with virtual circuits, but Frame Relay can be adapted for datagrams too.

   Since Frame Relay can have relatively large frame sizes, it can be used to encapsulate TCP/IP traffic although some changes to the protocol/ network may be required. For cell relay, carrying TCP/IP traffic is much more complicated, and the network must be faster, has to support fast switching and ATM protocol/services.

8. What is the difference between switching and routing? Are routing algorithms specific to a switching technology?

   Answer:
   Switching is the technology responsible for directing data traffic from one network path to another (typically at a point where many such paths are possible) according to established path or destination address and routing rules. Examples are: packet switching, where each data packet is switched individually along each switch/router node according to destination address or established virtual path, and circuit switching, where each switch is reserving a path for traffic carrying the circuit identifier.

   Routing is the methodology by which switches/routers are to decide how to route packets or establish circuits. Routing algorithms are responsible for finding efficient routes for messages

in the network, avoiding cycles, supporting bandwidth, and avoiding congestion. Routing algorithms can be XY-routing, worm-hole routing, OSPF routing, and others. As such, routing is specific to a switching technology, but on occasions it can be applied to different switching technologies or at least adapted to it.

9. How many sublayers are there in ATM? What are they?

   Answer:
   There are four sub-layers in ATM. In the AAL layer the sub-layers are the Convergence sub-layer (CS) and the Segmentation and Reassembly layer (SAR), where CS provides the interface for software to send/recieve data from SAR sub-layer, in-charge of breaking data into ATM cells, or merging cells for program data. There are another two sub-layers in the ATM Physical layer, and they are the Transmission Convergence (TC) sub-layer and the Physical Medium Dependent (PMD) sub-layer. TC puts the cells in the proper format for transmission and additionally does the error checking, while PMD is responsible for the physical data transmission specifications.

10. In HFC cable networks, two modulation schemes are used for sending downstream and upstream data. Why should the upstream case be handled differently from downstream? Should we employ different multiplexing technologies as well?

    Answer:
    Since HFC cable networks have many homes (clients) to one server in a broadcast network, the server can send data downstream in a synchronized manner and only the intended station will listen to the data, while others throw it away. There will be no collisions in this manner, and a multiplexing scheme such as FDM or TDM is sufficient to separate different logical transmission links. In the upstream however it is not so easy. Each client can asynchronously send data to the network and it's important to avoid collisions. Although an FDM multiplexing technology in the upstream will do, there might not be enough bandwidth for all users if channel allocations were fixed. On the other hand, TDM is hard to synchronize among distributed clients.

    In the downstream throughput is important, and since there's larger bandwidth, retransmission is not a problem and so QAM is an efficient modulation scheme to use. However, in the upstream, it is important to try to avoid retransmissions as the bandwidth is narrow and already congested with many clients trying to send data asynchronously, so an error resilient modulation scheme such as QPSK is required.

Chapter 16

Multimedia Network Communications and Applications

Exercises

1. Discuss at least two alternative methods for enabling QoS routing on packet-switched networks based on a QoS class specified for any multimedia packet (this would apply to any *store-and-forward* network).

   Answer:
   The packets in the network all have a class specifier to indicate the level of QoS required by the users for the specific payload.

   The first possible routing methodology is to have routers sort the packets as they arrive according to priority determined jointly by QoS required and the amount of time in the queue. The packets will be routed to their destination in sorted order with the first packet in the queue dispatched first.

   Another possible routing method though less likely is to classify network trunks according to low/high capacity/delay/channel error and forward packets as they come but through routes that are appropriate to the QoS requested so that high bandwidth/low delay channels won't get bogged down with low priority traffic. Such routing information can be maintained dynamically and at every router in the network.

2. Suggest a few additional priority delivery methods for specific multimedia applications that were not mentioned in Section 16.1.3.

   Answer:
   Some examples of additional priority delivery methods can be:

   - **Prioritization for Zerotree wavelet images. In a Zerotree representation the bits are sorted by bit-planes with higher importance bit-planes coded first and then lower ones. Each bit-plane can be given different priority for transmission, and thus achieve graceful degradation.**
   - **Prioritization of cellular services. Some of the cellular services might involve voice calls, internet browsing, online gaming, text messaging and e-mail. Since cell networks typically have a limited throughput especially on the wireless channels, prioritization of services is very important here. Of least importance is the text messaging and e-mail. This doesn't**

mean the packets are dropped, only sent when bandwidth is available and received likewise, there are no real-time requirements or guarantees on that service. Voice calls should get the highest priority since they are most important on a cell phone. Data calls and online gaming can receive some priority in between the other services depending on mainly market considerations.

- **Prioritization of services in collaborative environment. In a collaborative environment the most important element is the interaction among different people (over a network). The highest priority should be given thence to text messaging as it requires only few bytes and users expect it to be interactive, and nearly as high a priority with only minor delays for audio and then video, ideally synchronized. Of lower priority are the shared programs, such as slides, whiteboard, games, and yet the lowest priority are the non real-time services such as file sharing.**

3. When should RTP be used and when should RTSP be used? Is there an advantage in combining both protocols?

   Answer:
   The RTP protocol is used to help the decoder/renderer play a media in the correct speed, which is even more essential when the coded media structure is adaptive (such as adaptively put I-frames in a video where there's a scene change, and add more B-frames where there's motion), and to synchronize multiple streams together so that audio/video are played together at the right time. It is mainly used over Multicast channels.

   RTSP is used to control a media stream on the server. It allows connection to the media, requesting streaming, VCR-like functionality, and other extensions as necessary by the application. RTSP will be mainly used over Unicast connections since the VCR functions should not control the server in a Multicast. However it is still possible to use it on Multicast for the other functions it provides.

   It is indeed useful to combine both protocols since a server streaming media would most of the time need to support both Multicast and Unicast streams, and moreover the two protocol complement each other; while a client requests a stream via RTSP and controls it, the renderer would use RTP to keep the video synchronized with the audio and not lagging even in the presence of packet loss.

4. Consider again Figure 16.4 illustrating RSVP. In (d) receiver R3 decides to send an RSVP RESV message to S1. Assuming the figure specifies the complete state of the network, is the path reserved optimal for maximizing future network throughput? If not, what is the optimal path? Without modifying the RSVP protocol, suggest a scheme in which such a path will be discovered and chosen by the network nodes.

   Answer:
   Although the path that R3 reserves is the one sent by the PATH message from S1, it reserves 2 unused links before the stream merges at node A. A more optimal path (assuming equal link capacity naturally) would be R3 to C and merge there, thus reserving only one additional link and saving bandwidth for future streams.

   A more optimal path could be established by the RSVP servers when they send the PATH message. A possible scheme can be to choose a link with highest remaining capacity as the main trunk for transmission and branch off from it to clients at points that minimize a measure combining the number of hops to the client and the capacity of such links. The idea would be to allow as many as possible future reservations.

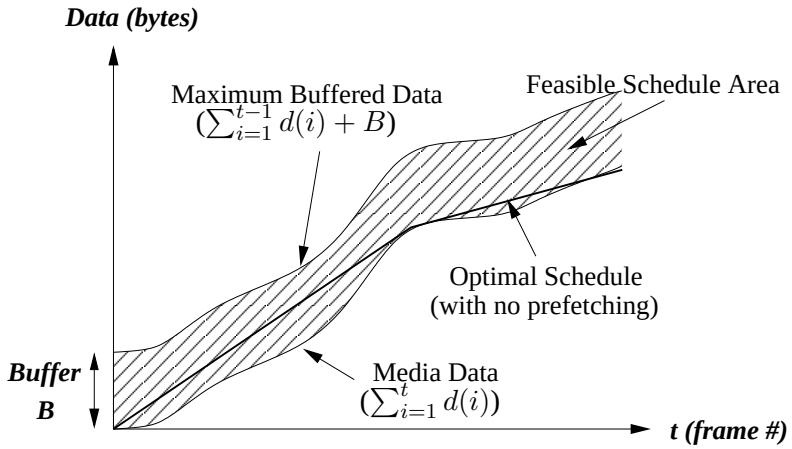


Fig. 16.17: (Exercise-answer) The grayed out area is the space where all feasible transmissions can exist, while the solid line is the only optimal transmission schedule to minimize rate variability without prefetching any data first.

5. Browse the web to find current technologies designed for Internet telephony.

   Answer:
   Use Google or other search engines to look-up Internet Telephony. e.g. *http://www.iptel.org* provides a list of products from servers and routers to diagnostic software. Also there's an Internet Telephony magazine at *http://www.tmcnet.com/it/*.

6. For Staggered broadcasting, if the division of the bandwidth is equal among all K logical channels ($K \geq 1$), show that access time is independent of the value of K.

   Answer:
   Let M be the number of movies, L the length of each movie, and B the bandwidth of the link, then the access time for any movie is $\delta = \frac{M \cdot L}{B}$ which is independent from K.

   If $K = 1$, it is a simple round-robin. When $K > 1$, the movies are staggered — customer can get the movie from the next available (logical) channel. However, since the same total bandwidth B now had to be divided among K logical channels, the access time is not improved.

7. Specify on Figure 16.17 the characteristics of *feasible* video transmission schedules. What is the *optimal transmission schedule*?

   Answer:
   See Fig.16.17 (Exercise-answer).

8. For the optimal work-ahead smoothing technique, how would you algorithmically determine at which point to change the planned transmission rate? What is the transmission rate?

   Answer:
   Suppose we are looking at frame q of the video. We know the segment start point is frame $p+1$. We can detect overflow when the minimum possible rate R_{min} over the interval $p + 1 - q$ is higher than the rate to totally fill the buffer at time q, that is when:

$$R_{min} > \frac{W(q) - (D(p) + B(p))}{q - p}$$

Note: **at the last frame we have nothing more to buffer, hence is $q = N$ we also consider it the end of the segment and use R_{min} over it. Since an overflow occured at minimum rate, we must reduce rate at the last point q' where R_{min} was achieved (was exactly equal to it). The new rate segment is then from frames $p + 1 - q'$ at rate R_{min}. We set $p = q'$ for the next segment.**

Similarly, to know where an underflow occured we test for

$$R_{max} < \frac{D(q) - (D(p) + B(p))}{q - p}$$

If indeed the maximum possible rate over the interval is less than the minimum necessary to play the stream then we must increase the rate at last frame q' where R_{max} was achieved. The transmission rate for the segment is naturally R_{max}.

9. Considering again the optimal work-ahead smoothing technique, it was suggested that instead of using every video frame, only frames at the beginning of statistically different compression video segments can be considered. How would you modify the algorithm (or video information) to support that?

   Answer:
   Here, we must assume the video is coded in a way that finds significant points of compression change, presumably scene changes or larger motion, and so on. Let's assume the compressed video has a way of indicating this for example by assigning new I-frames to only such frames or starting a new GOP every time a the coder adapts to a new sequence. We can now create a new virtual frame sequence S' where the frames are only the ones indicated by the compressed video as a point of change. However, instead of specifying the frame size as in the source video, the frame size is the total amount of data in the video up to the frame (including the frame) minus the previous frame size in the virtual sequence. This gives only the amount of data needed to transmit up to the compression change frame, and we can safely assume that this data can be transmitted at constant bit-rate, whatever rate it is as calculated by the work-ahead smoothing algorithm.

   The technique is applied to sequence S' instead of the original video and so does not need any other modification.

10. *Unicast* transmission is when a server establishes a single communication channel with a specific client.

    (a) For video streaming, suggest a couple of methods for unicast video transmission, assuming the video is VBR encoded and client feedback is allowed.

    (b) Which one of your methods is better? Why?

    Answer:
    One method is to calculate the optimal work-ahead plan for the video and transmit it to the client according to the plan. The only feedback necessary is for retransmissions and those take overhead bandwidth on top of the planned rate.

    A second method is to always try to keep the client work-ahead buffer full. A client will send feedback as to its available space in the buffer and the server will stream media as fast as the link allows to fill the buffer. Retransmissions are handled similarly to the first method, although here they have to take a bandwidth slice from the media stream.

    On the whole keeping the work-ahead buffer always full is a better approach for the simple reason that an unreliable network can get congested at any point in time and a full buffer will allow

the network the highest possible time for recovery before the end-user experiences playback problems. However there are many reasons why this method is not so effective either. First, for practical/commercial reasons we would like to limit the bandwidth to the smallest amount possible, so that we can stream more movies over the same link. Also, feedback is not always reliable, and if not arrived in time it could cause buffer overflow, or less likely underflow. It's important to avoid those for the reasons given in the text. A good approach would be to combine the work-ahead plan with an attempt to keep the buffer full. The reserved rate is the peak rate of the plan, and we will always try to stream at that rate unless the client buffer is full (will use a lower rate to keep it full). If there is no feedback at the time expected, then we can calculate the expected feedback from the last feedback and the plan. Naturally this can be wrong as well but it has higher chance for avoiding overflow/underflow.

11. *Multicast* transmission is when a server transmits a single multimedia stream to all listening multicast routers, and they forward it until a client receives the stream.

    (a) For VBR video streaming, suggest a couple of methods for multicast video transmission.

    (b) Which one of your methods is better? Why?
    Hint: Although a client may have a reverse channel for feedback, if all clients send feedback, it would cause congestion in the network.

    Answer:
    The first possible method is once more to calculate the optimal work-ahead plan for the video and to try to stream according to it. Error correction is probably done using Forward Error Correction schemes.

    Another possible approach is to assign one client as a feedback client and make it report the status of the work-ahead buffer in the name of all the clients. Once again the server will try to keep the buffer full at all time.

    Although it is still a good idea to keep the buffer full since it maximizes the allowable recovery time is the network becomes congested, the Multicast case is worse than the Unicast case since we really don't know the state of all the clients' buffers, and even if we did we couldn't accommodate them all had they been different. Different links on different parts of the Multicast network can be slower or faster and clients can be joining and leaving at any time, so having only some representative feedback clients is not easy to achieve. In this scenario we can still adapt the streaming technique described in the previous question for combining the work-ahead plan with trying to keep the buffer full. In this case it could be reasonable to not take any client feedback and just calculate the amount of data that can still be streamed to the client work-ahead buffer. The approach would be to fill the buffer with data as fast as possible yet without streaming faster than the calculated plan peak-rate.

12. Suggest how you would encode an H.261 video (described in Chapter 10) to use a CBR stream with rate equal to the channel bandwidth.

    Hint: use the input/output frame buffer and quantization control in the encoder and decoder loops.

    Answer:
    In the encoder, the output buffer will hold a fixed amount of data. For CBR stream a fixed amount of data needs to be streamed in a a fixed amount of time. Define a window w over the output buffer, then require that within that window a full second of video would be encoded. Hence, this is encoder rate control by the second. We can collect the proper number of frames

in a second (according to their timestamp), if a frame duration is consistent it can be something like 24 frames, let's call that the window w. At this point we have a fixed amount of bits b we are allowed to allocate to the frames in w. If the encoded frames take less bits than b we can increase quality by the difference by requantizing these frames with a smaller quantizer. If they take more bits than b then we have to use a larger quantizer.

This might seem simple, but it is actually very involved. Among the numerous frames in the buffer and the many GOBs for each frame, we have to decide where it is best to assign the available bits by reducing quantization size there. There are many possible schemes for deciding where to allocate more or less bits. Most of them have to do with Human Psychovisual response, e.g. that we notice less detail in motion, that we notice less detail in darker areas, and that we are more sensitive to low frequency discontinuities across MB boundaries. At the same time, we might want to give more bits to I-frames, somewhat less to P-frames, and very little to B-frames, since I-frames would be used to predict from and hence their quality is more important.

Chapter 17

Wireless Networks

Exercises

1. In implementations of TDMA systems such as GSM and IS-136, and to a lesser degree in networks based on CDMA, such as IS-95, an FDMA technology is still in use to divide the allocated carrier spectrum into smaller channels. Why is this necessary?

   Answer:
   It is necessary to divide the allocated spectrum due to device and physical limitations. When the time interval gets too small the mutipath fading effect can make the signal interfere with other time intervals at the base station. The electronics also generate thermal noise and can lack the precision to separate signals at very short time intervals. For CDMA, the device limitation is in encoding and transmitting a large enough chip code in a second.

2. Discuss the difference between the way GSM/GPRS and WCDMA achieve variable bitrate transmissions.

   Answer:
   GSM/GPRS uses TDMA mainly. GPRS, using a slotted reservation scheme, where normally each cellular phone (mobile station) has one time slot for uplink and one for downlink, can assign more than one slot for a particular mobile requiring higher data rate. Naturally, this takes away from the ability of other mobiles to transmit at higher rates (reserve more slots).

   WCDMA achieves variable bit-rates by adapting variable chip code length (Orthogonal Variable Spreading Factor codes - OVSF), since the chip rate is fixed, with shorter codes we can send more data in the 10ms frame, unfortunately at the expense of lower processing gain (lower stability). Even worse, there are so few such codes that they cannot be unique among terminals (consider a Spreading Factor of 4, that means only 4 codes exist!). This is the reason PN scrambling codes are used to separate different mobiles. There are millions of such codes and the spreading factor for scrambling is exactly 1. Eventually, to allow even higher data-rates, a mobile is allow to have a few different scrambling codes and transmit to them all at once. This is similar to GPRS in some ways.

3. We have seen a geometric layout for a cellular network in Figure 17.1. The figure assumes hexagonal cells and a symmetric plan (i.e., that the scheme for splitting the frequency spectrum over different cells is uniform). Also, the reuse factor is $K = 7$. Depending on cell sizes and radio interference, the reuse factor may need to be different. Still requiring hexagonal cells, can all possible reuse factors

achieve a symmetric plan? Which ones can? Can you speculate on a formula for general possible reuse factors?

Answer:
Not all possible reuse factors can achieve a symmetric plan, for example when the reuse factor is $K = 2$ there is no symmetric plan since one of the cells must border a cell with the same frequencies, while the other does not. The possible reuse factors are 1, 3, 4, 7, 9, 12, ... and the formula is:

$$K = (i + j)^2 - i \cdot j, \qquad i, j = 0, 1, 2, 3, ...$$

4. What is the spreading gain for IS-95? What is the spreading gain for WCDMA UTRA FDD mode, assuming all users want to transmit at maximum bitrate? What is the impact of the difference between the spreading gains?

   Answer:
   The Spreading Factor (SF) is defined as W/R where W is the bandwidth and R is the bit-rate, and the processing gain is typically described in dB. For IS-95 we know the bandwidth is $1.25MHz$ and for WCDMA FDD mode it's $5MHz$. However, technically it's not accurate to take the bandwidth into account in practical CDMA systems since the transmit bandwidth is essentially equal to the chip-rate (note that this implies the number of chips in the channelization sequence is equal to the SF as described in the chapter).

   The SF for IS-95 is equal to $SF = W/R = 1.2288M/9.6K = 128$, where W is equal to the IS-95 chip-rate and R here is the constant rate of the voice codec. The processing gain is therefore $PG = 10log_{10}(128) \approx 21$.

   For WCDMA UTRA FDD mode the Spreading Factor ranges from $4 - 512$ in the downlink frequencies, and $4 - 256$ in the uplink. For maximum bit-rate, a SF factor of 4 is used. The processing gain for that is $PG = 10log_{10}(4) \approx 6$. In an IS-95 system the channel noise is assumed to be a nominal $7dB$, so we see that special conditions are necessary for WCDMA to use a $SF = 4$. In particular the environment has to have low noise and we have to use half-rate transmission with 3 parallel scrambling codes in the downlink and 6 in the uplink. Calculating the data-rate in this scheme we have $R = 6 \cdot \frac{1}{2}\frac{3.84M}{4} = 2.88M$, though the half-rate data transmission has some overhead and is rated at about maximum $2.3Mbps$.

5. When a cellular phone user travels across the cell boundary, a *handoff* (or handover) from one cell to the other is necessary. A hard (imperfect) handoff causes dropped calls.

   (a) CDMA (Direct Sequence) provides much better handoff performance than FDMA or Frequency Hopping (FH). Why?

   (b) Suggest an improvement to handoff so it can be softer.

   Answer:

   (a) FDMA and FH have a reuse factor larger than 1, i.e., the neighboring cell is using a different frequency. CDMA uses the same frequency band for all cells, so handoff is much easier.

   (b) Allow the handset to communicate with more than one cell boundary simultaneously, so handoff can be smoother, less chance of dropping calls. This is actually used in CDMA (adapting different spreading code for each if necessary).

6. In a CDMA cell, when a CDMA mobile station moves across a cell boundary, a *soft handoff* occurs. Moreover, cells are also split into sectors, and when a mobile station moves between sectors, a *softer handoff* occurs.

   (a) Provide arguments for why a softer handoff is necessary.

   (b) State at least one other difference between the two handoffs.

   Hint: During handoff in a CDMA system, the mobile stations can transmit at lower power levels than inside the cell.

   Answer:

   (a) At every sector the base station assigns Orthogonal or PN codes to the mobile stations. These codes need to be consistent for the sector and might be different for the same mobile station in different sector. It is necessary for the mobile to acquire the new spreading code to use as it crosses sector boundaries, hence a softer handoff is necessary.

   (b) Another difference between the soft and softer handoffs is that in a soft handoff between cells the mobile can transmit at lower power since two base station hear it through two distinct fading paths. In the softer handoff case, the same fading path is traversed by the signal to the same base station. The only difference is the directional antennas but this amounts to tuning Rake receivers at any particular antenna, so it is not likely that the mobile needs to transmit at much lower power levels.

7. Most of the schemes for channel allocation discussed in this chapter are fixed (or uniform) channel assignment schemes. It is possible to design a dynamic channel allocation scheme to improve the performance of a cell network. Suggest such a dynamic channel allocation scheme.

   Answer:
   The idea would be to keep dynamic stats in a centralized way (across many cells) about how many mobiles are trying to access the network at any particular cell at any time. The more mobiles accessing it the more channels should be allocated there and taken away from the least used neighboring cell which can recursively take channels away from its neighbors. Possible algorithms to achieve it can use dynamic programming and Reinforced Learning which is argued to be even more efficient. The interested reader should refer to recent papers in the matter.

8. The 2.5G technologies are designed for packet-switching services. This provides data-on-demand connectivity without the need to establish a circuit first. This is advantageous for sporadic data bursts.

   (a) Suggest a method to implement multiple access control for TDMA packet services (such as GPRS).

   (b) Circuits are more efficient for longer data. Extend your suggested method so that the channel goes through a contention process only for the first packet transmitted.

   Hint: Add reservations to your scheme.

   Answer:

   (a) We can use a Slotted Aloha method for sending data in available time slots. That is instead of or in addition to the single uplink channel assigned to each mobile station, each station

can contend for more slots for increased data rates. In the Slotted Aloha scenario if a mobile has more data to transmit it will pick up a time slot at transmit on it. If it detects a collision (or informed of it by the base station), it will queue the packet for retransmission and wait a random number of time slots before trying to transmit again (now a longer message most likely).

(b) One problem with the suggested method above is that since each mobile can take over any time slot, it needs to identify itself to the base station at every time slot, which is an overhead. We can reduce the overhead if we use Reservation protocol like Reservation Aloha that assigned a particular time slot to a particular mobile both at the mobile and base station, after it has been acquired for the first time. A slot is acquired when a successful transmission occurs. Every mobile has to listen to all time slots and is allowed to transmit only in slots it didn't detect traffic in the previous frame. This way there are fewer collisions (no collisions for all frames other than possibly the first). There is also no need to retransmit the mobile id since the mobile using the time slot is known.

9. H.263+ and MPEG-4 use RVLCs, which allow decoding of a stream in both forward and backward directions from a synchronization marker. The RVLCs increase the bitrate of the encoding over regular entropy codes.

   (a) Why is this beneficial for transmissions over wireless channels?

   (b) What condition is necessary for it to be more efficient than FEC?

   Answer:

   (a) Over wireless channels there is noise that can produce random bit errors, and we would like to avoid retransmission as bandwidth is more limited and most applications are real time. It is important then to use error resilient coding at the expense of coding efficiency.

   (b) It will be more efficient than FEC where the channel can have many errors and thus requires many FEC bits, and yet at sporadic times, hence most of the time the FEC is useless overhead. This scenario can be quite common in wireless channels.

10. Why are RVLCs usually applied only to motion vectors? If you wanted to reduce the bitrate impact, what changes would you make?

    Answer:
    RVLCs are not applied to DCT coefficient since these are view as much less important if are lost and easier to estimate than MVs. MVs are essential at most frames for reasonable visual quality, the residuals are not as important, and although I-frames can be treated differently, there aren't that many I-frames in a video sequence comparatively.

    To reduce the bit-rate impact we can apply RVLCs only on the headers and not protect the MVs as well. Naturally, this will increase chances of annoying visual defects in the presence of noise.

Chapter 18

Content-Based Retrieval in Digital Libraries

Exercises

1. What is the essence of *feature localization*? What are the pros and cons of this approach, as opposed to the traditional image segmentation method?

   Answer:
   The essence of *feature localization* is to advocate an approach in which useful visual features are identified according to their locality. This is based on the observation that precise image segmentation of generic images/videos (noisy and ambiguous) is unattainable. Instead, a process of determining statistical support of some feature (presumably homogeneous in the object) is fast and more appropriate to use, since in image and video processing and analysis, precise pixel membership is hardly ever required. Feature localization assumes pixels of the same feature in a tile belong to the same object. Tiles give initial object-based statistics to merge on and grow the locale. A group of feature locales forms an object.

   In traditional image segmentation methods, each individual pixel membership matters and since initial level statistics (for a pixel) are not known, it invites the use of regression methods which are slower and could have problems converging to the right membership groups. The advantage of localization is that it is faster and more accurate in convergence with the premise that exact membership is not important, whereas in traditional segmentation techniques slight pixel errors might lead to incorrect convergence. The disadvantage of the localization scheme is that it lacks exact shape definition for the locale, therefore shape processing (extracting/matching/coding) will be harder to do than in the image segmentation scenario.

2. Show that the update equation (Eq. 18.9) is correct — that is, the eccentricity $E_j^{(k+1)}$ for parent locale j at iteration $k+1$ can be derived using the eccentricity, centroid, and mass information for the parent locale j and child locale i at iteration k. (Note: $C_{x,j}^{(k)}$ and $C_{y,j}^{(k)}$ are the x and y components of the centroid $\mathbf{C}_j^{(k)}$, respectively.)

   Answer:
   From the Feature Localization section in the Chapter we have the eccentricity equation for

locale j:

$$\begin{aligned} E_j &= \sum_{m=1}^{M_j} \frac{\|\mathbf{P}_m - \mathbf{C}_j\|^2}{M_j} = \sum_{m=1}^{M_j} \frac{(P_{x,m} - C_{x,j})^2 + (P_{y,m} - C_{y,j})^2}{M_j} \\ &= \sum_{m=1}^{M_j} \frac{P_{x,m}^2 + P_{y,m}^2}{M_j} + \sum_{m=1}^{M_j} \frac{C_{x,j}^2 + C_{y,j}^2}{M_j} - \sum_{m=1}^{M_j} \frac{2P_{x,m}C_{x,j} + 2P_{y,m}C_{y,j}}{M_j} \\ &= \sum_{m=1}^{M_j} \frac{P_{x,m}^2 + P_{y,m}^2}{M_j} + \frac{M_j C_{x,j}^2 + M_j C_{y,j}^2}{M_j} - 2C_{x,j}^2 - 2C_{y,j}^2 \\ &= \sum_{m=1}^{M_j} \frac{P_{x,m}^2 + P_{y,m}^2}{M_j} - C_{x,j}^2 - C_{y,j}^2 \end{aligned}$$

So we can calculate $E_j^{(k+1)}$ for parent Locale j and child Locale i as follows:

$$\begin{aligned} E_j^{(k+1)} &= \sum_{m=1}^{M_j^{(k+1)}} \frac{{P_{x,m}^{(k+1)}}^2 + {P_{y,m}^{(k+1)}}^2}{M_j^{(k+1)}} - {C_{x,j}^{(k+1)}}^2 - {C_{y,j}^{(k+1)}}^2 \\ &= \frac{\sum_{m=1}^{M_j^{(k)}} {P_{x,m,j}^{(k)}}^2 + {P_{y,m,j}^{(k)}}^2 + \sum_{m=1}^{M_i^{(k)}} {P_{x,m,i}^{(k)}}^2 + {P_{y,m,i}^{(k)}}^2}{M_j^{(k+1)}} - {C_{x,j}^{(k+1)}}^2 - {C_{y,j}^{(k+1)}}^2 \\ &= \frac{M_j^{(k)}(E_j^{(k)} + {C_{x,j}^{(k)}}^2 + {C_{y,j}^{(k)}}^2) + M_i^{(k)}(E_i^{(k)} + {C_{x,i}^{(k)}}^2 + {C_{y,i}^{(k)}}^2)}{M_j^{(k+1)}} - {C_{x,j}^{(k+1)}}^2 - {C_{y,j}^{(k+1)}}^2 \end{aligned}$$

3. Try the VIPER search engine, refining the search with relevance feedback for a few iterations. The demo mentions Gabor histograms and Gabor blocks. Read enough of the files associated with the site to determine the meaning of these terms, and write a short explanation of their use.

Answer:
"Design and Evaluation of a Content-Based Image Retrieval System", in "Design and Management of Multimedia Information Systems: Opportunities and Challenges by Syed Mahburbur Rahman (ed.), Idea Group Publishing, 2001, states that "Viper uses a palette of 166 colours, derived by quantizing HSV space into 18 hues, three saturations, three values and four grey levels. Two sets of features are extracted from the quantized image. The first is a colour histogram, where empty bins are discarded. The second represents colour layout. Each block in the image (the first being the image itself) is recursively divided into four equal-sized blocks, at four scales. The occurrence of a block with a given mode colour is treated as a binary feature. There are thus 56,440 possible colour block features, of which each image has 340."

Then the Gabor features are defined as a bank of real, circularly symmetric Gabor filters defined via

$$f_{mn}(x,y) = \frac{1}{2\pi\sigma_m^2} e^{-\frac{x^2+y^2}{2}\sigma_m^2} \cos\left(2\pi\left(u_{0m}x\cos(\theta_n) u_{0m}y\cos(\theta_n)\right)\right).$$

The authors state that "m indexes filter scales, n their orientations and u_{0m} gives the center frequency. The half peak radial bandwidth is chosen to be one octave, which determines σ_m. The highest center frequency is chosen as $u_{01} = 0.5$, and $u_{0m+1} = u_{0m}/2$. Three scales are used.

The four orientations are: $\theta_0 = 0$, $\theta_{n+1} = \theta_n + \pi/4$. The resultant bank of 12 filters gives good coverage of the frequency domain, and little overlap between filters. The mean energy of each filter is computed for each of the smallest blocks in the image. This is quantized into 10 bands. A feature is stored for each filter with energy greater than the lowest band. Of the 27,648 such possible features, each image has at most 3,072. Histograms of the mean filter outputs are used to represent global texture characteristics."

In the Relevance Feedback mechanism, "The weighting used is based on the calculations of the term frequency tf_j (frequency of feature j in the image) and the collection frequency cf_j (frequency of the feature j in the entire database) of the feature, as well as its type (block or histogram)."

4. Try a few of the more experimental image search engines in the Further Exploration section above. Some are quite impressive, but most are fairly undependable when used on broad data domains.

   Answer:
   The search term "dinosaur", in Google Image Search, is very good at producing over 50,000 good dinosaur images. However, a broader search, say for an emption such as "fear" produces many more hits, basing targets mostly on image filenames and thus hitting many humans named "Fear", etc.

   A search on "depression" gives some 40,000 images ranging from the human condition (the objective, here) to a vast array of images of the 1930s to holes in the ground.

5. Devise a text-annotation taxonomy (categorization) for image descriptions, starting your classification using the set of Yahoo! categories, say.

   Answer:
   Yahoo has a top-level taxonomy of 13 classes, such as Art, Entertainment, Finance, News, Sports, etc. Within sports.yahoo.com, there is a sub-taxonomy of a further 9 classes (Baseball, Basketball, Football, Hockey, Soccer, MoreSports, Sailing, OtherSports, Pick'emGames), and of course a finer classification within these.

6. Examine several web site image captions. How useful would you say the textual data is as a cue for identifying image contents? (Typically, search systems use *word stemming*, for eliminating tense, case, and number from words — the word *stemming* becomes the word *stem*.)

   Answer:
   The caption is most important in combination with the image contents:

   > **"The period following the split announcement, the Post-Announcement stage, often sees a *depression*."**

   However, in "*Construction of a Hierarchical Classifier Schema using a Combination of Text-Based and Image-Based Approaches", Cheng Lu and Mark S. Drew, ACM SIGIR 2001, The 24th Annual Conference on Research and Development in Information Retrieval, pp.438-439, New Orleans, September 9-13, 2001,*
   http://www.cs.sfu.ca/∼mark/ftp/Sigir01/sigir01.pdf
   we showed that performing classification on a hierarchy differently on different levels of the tree, using text for branches and images only at leaves, improved web document classification performance significantly.

7. Suggest at least three ways in which audio analysis can assist in video retrieval-system-related tasks.

Answer:
Firstly, we note that while the video in a scene can contain significant alterations, even cuts, whie a scene is playing out usually the audio remains smooth throughout. That is, one of the salient features of a *scene* is that audio is smooth, even if video is not.

Secondly, important aspects of a human's speaking voice can be utilized. For example, in a lecture, usually the frequency raises at the beginning of a new topic.

Thirdly, the characterisitics of human speech itself can be used to determine whether people are present in a scene, along with color and other video features.

8. Suppose a color histogram is defined coarsely, with bins quantized to 8 bits, with 3 bits for each red and green and 2 for blue. Set up an appropriate structure for such a histogram, and fill it from some image you read. Template Visual C++ code for reading an image is on the text web site, as `sampleCcode.zip` under "Sample Code".

Answer:
Here we shall simply give a `matlab` version:

```
% rgbhist.m
% matlab script
im=imread('resources_exercises/chap18/lena256.jpg');
ss = size(im); % rows x cols x 3
% let's just truncate values to 3-bits,3-bits,2-bits for R,G,B:
im = double(im);
imR = im(:,:,1);
imG = im(:,:,2);
imB = im(:,:,3);
scaleR = 256/8; % 3-bit
scaleG = 256/8; % 3-bit
scaleB = 256/4; % 2-bit
imR =  fix(imR/scaleR)*scaleR;
imG =  fix(imG/scaleG)*scaleG;
imB =  fix(imB/scaleB)*scaleB;
im(:,:,1) = imR;
im(:,:,2) = imG;
im(:,:,3) = imB;
%
values8 = unique(fix(0:255/scaleR)*scaleR);
% 0 32 64 96 128 160 192 224
values4 = unique(fix(0:255/scaleB)*scaleB);
% 0 64 128 192
%
im = reshape(im, ss(1)*ss(2), 3);
hist332 = zeros(8,8,4);
for i=1:8
  for j=1:8
    for k=1:4
      temp =  (im(:,1)==values8(i)) & ...
        (im(:,2)==values8(j)) & ...
```

```
        (im(:,3)==values4(k));
      % size(temp) = ss(1)*ss(2)
      hist332(i,j,k) = sum(temp);
    end
  end
end
sum(sum(sum(hist332))) % ==ss(1)*ss(2)==number of pixels.
plot3(hist332(:,:,1),hist332(:,:,2),hist332(:,:,3),'.');
```

9. Try creating a texture histogram as described in Section 18.2.5. You could try a small image and follow the steps given there, using MATLAB, say, for ease of visualization.

   Answer:
   Here we simply give the beginning steps: creation of a good edge map.

```
% texhist.m
% matlab script
im = imread('resources_exercises/chap18/lena256.jpg');
im = rgb2ycbcr(im);
im = im(:,:,1); % luminance
im = double(im);
ss = size(im);
rows = ss(1);
cols = ss(2);

% edge magnitude
sobelx = [[-1 0 1] ;
          [-2 0 2] ;
          [-1 0 1]];
sobely = sobelx';
imdx = conv2(im,sobelx,'same');
imdy = conv2(im,sobely,'same');
D = sqrt(imdx.^2 + imdy.^2);
% edge direction
phi = atan2(imdy,imdx); % in -pi .. pi
phi = phi*180/pi; % in -180 .. 180
codes = direction_code(im);

% non-maximum suppression:
mask = ones(ss); % edge pixels not to discard
% first, don't use border pixels:
mask(:,1) = 0; mask(:,end) = 0; mask(1,:) = 0; mask(end,:) = 0;
for i=2:(rows-1) % avoid borders
  for j=2:(cols-1)
    switch codes(i,j)
      case 0  % horiz
  i1 = i; j1 = j-1; i2 = i; j2 = j+1;
      case 1  % centered on 45 degrees
```

```
  i1 = i+1; j1 = j-1; i2 = i-1; j2 = j+1;
      case 2  % vertical
  i1 = i-1; j1 = j; i2 = i+1; j2 = j;
      case 3  % centered on 135 degrees
  i1 = i-1; j1 = j-1; i2 = i+1; j2 = j+1;
    end
    %
    mag = D(i,j);
   % has larger neighbor pixel close in direction
    if ~( ...
   ( mag < D(i1,j1) ) && (~largeanglediff(phi(i,j),
   phi(i1,j1)) ) ...
         ||  ...
         ( mag < D(i2,j2) ) && (~largeanglediff(phi(i,j),
          phi(i2,j2)) ) ...
        )
      % suppress, since not (local max And same direction):
      mask(i,j) = 0;
    end
  end
end

% mask is local-maximum pixels And D>Threshold
% Choose Thresh as excluding bottom 10% of D values:
[counts,bincenters] = hist(D(:));
thresh = mean(bincenters(1:2));
temp = (D(:)>thresh);
mask2 = reshape(temp,ss);
mask = mask & mask2;

%remove isolated pixels: if 3x3 around pixel is surrounded by ~mask:
mask3 = ones(ss);
for i=3:(rows-2)
  for j=3:(cols-2)
    if (     mask(i,j) &&  ...
              ...
             ~mask((i-2),(j-2)) && ...
             ~mask((i-2),(j-1)) && ...
             ~mask((i-2),j) && ...
             ~mask((i-2),(j+1)) && ...
             ~mask((i-2),(j+2)) && ...
              ...
             ~mask((i-1),(j-2)) && ...
             ~mask((i-1),(j+2)) && ...
              ...
             ~mask(i,(j-2)) && ...
             ~mask(i,(j+2)) && ...
              ...
```

```
              ~mask((i+1),(j-2)) && ...
              ~mask((i+1),(j+2)) && ...
               ...
              ~mask((i+2),(j-2)) && ...
              ~mask((i+2),(j-1)) && ...
              ~mask((i+2),j) && ...
              ~mask((i+2),(j+1)) && ...
              ~mask((i+2),(j+2)) ...
       )
       mask3(i,j) = 0;
    end % if
  end
end

mask = mask & mask3;
myimwrite1(mask+0.0,'mask.jpg');

D = D.*mask;
%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% largeanglediff.m
function bool = largeanglediff(a1,a2)
% more than 45 degrees is "large" difference
temp = abs(a1-a2);
if temp>180
  temp = 360-temp;
end
if temp>45
  bool = true;
else
  bool = false;
end
```

```
% direction_code.m
function code = direction_code(angle)
% Make angle into an int==code for four directions,
% centered on 0, 45, 90, 135,
% independent of vector sense.
if angle > 180
  angle = 360 - angle;
end
code = mod( fix((angle+22)/45.0) , 4 );
```

The result of the above, for image `lena` is shown in Fig. 18.21.

Fig. 18.21: Edge mask.

10. Describe how you may find an image containing some 2D "brick pattern" in an image database, assuming the color of the "brick" is yellow and the color of the "gaps" is blue. (Make sure you discuss the limitations of your method and the possible improvements.)

    (a) Use color only.

    (b) Use edge-based texture measures only.

    (c) Use color, texture and shape.

    Answer:
    Use CBIRD, online, for this task.

11. The main difference between a static image and video is the availability of motion in the latter. One important part of CBR from video is motion estimation (e.g., the direction and speed of any movement). Describe how you could estimate the movement of an object in a video clip, say a car, if MPEG (instead of uncompressed) video is used.

    Answer:
    Since MPEG stores motion vector information (in compressed form), motion information is already readily available in the video without further processing being necessary. A simple approach to motion estimation is to adopt the assumption that most parts of the frame are relatively unchanging. Thresholding for higher values of the lengths of motion vectors gives a distribution of motion vectors for the moving object, and these can be used then to estimate a representative overall motion.

    If we wish to also take into account the "dominant motion" consisting mostly of camera motion, we can first carry out motion compensation by mapping subsequent frames back in time; here, we assume that most of the frame does not consist of moving objects, so that motion vectors mainly arise from camera zooming, panning, etc. Then the same thresholding of motion-

compensated frames and dominant-motion-compensated motion vectors can find the moving object.

12. Color is three-dimensional, as Newton pointed out. In general, we have made use of several different color spaces, all of which have some kind of brightness axis, plus two intrinsic-color axes.

    Let's use a *chromaticity* 2-dimensional space, as defined in Eq. (4.7). We'll use just the first two dimensions, $\{x, y\}$. Devise a 2D color histogram for a few images, and find their histogram intersections. Compare image similarity measures with those derived using a 3D color histogram, comparing over several different color resolutions. Is it worth keeping all three dimensions, generally?

    Answer:
    A few examples will suffice:

```
% matlab script
im = imread('resources_exercises/chap18/lena256.jpg');
ss = size(im);
im = reshape(im,prod(ss(1:2)),3);
ch = makechrom(im);
chshow(ch,ss(1),ss(2));
chhist = myhist2d(ch); %16 x 16 histogram
mesh(chhist);

% Let's show that cutting up the image makes no difference:
% flip top and bottom halves:
im = reshape(im,ss);
im2 = im; % declare
im2(1:(ss(1)/2),:,:) = im((ss(1)/2+1):end,:,:);
im2((ss(1)/2+1):end,:,:) = im(1:(ss(1)/2),:,:);
im = reshape(im,prod(ss(1:2)),3);
im2 = reshape(im2,prod(ss(1:2)),3);
ch2 = makechrom(im2);
chhist2 = myhist2d(ch2);
max(max(chhist2-chhist)) %  0


% Let's just make an image using a diagonal transform:
im = double(im);
im2 = im*diag([0.9, 1.1, 1.05 ]);
ch2 = makechrom(im2);
chshow(ch2,ss(1),ss(2));
chhist2 = myhist2d(ch2);
mesh(chhist2);
% histogram intersection:
%    closer image is neaer to 1.0 value.
intersec = histint2d(chhist,chhist2) % 0.7818
%
% and compare to a 3D hist:
hist3d = myhist3d(im);
hist3d2 = myhist3d(im2);
```

```
intersec = histint3d(hist3d,hist3d2) % 0.3527
% so 2D is considerably better than 3D.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%
% makechrom.m
function r=makechrom(mat)
% mat is Nx3
[rows,cols] = size(mat);
mat = double(mat);
denom = mat(:,1)+mat(:,2)+mat(:,3);
back = denom==0;
denom(back)=999;
r = zeros(rows,2);
for k=1:2
 tempr = mat(:,k)./denom;
 tempr(back)=0;
 r(:,k) = tempr;
end

%%%%%%%
% myimshow3.m
function fun(im3,r,c)
im3 = double(im3);
im3 = im3/max(max(im3));
temp=reshape(im3,r,c,3);
imshow(temp);

%%%%%%%
% chshow.m
function fun(ch,r,c)
% param is mxn x 2
%
% make a 3-d chrom, for display
ch3 = [ ch 1-ch(:,1) - ch(:,2) ];
myimshow3(ch3,r,c)


%%%%%%%
% myhist2d.m
function ahist = myhist2d(ch)
%param is mxn by 2; N is 16, say
  N = 16;
  ch = double(ch);
  ch = 255/max(max(ch))*ch;
  ahist = zeros(N,N) ;
```

```
  % Just take blacks as all-0 chs:
  blacks = (ch(:,1)==0) & (ch(:,1)==0);
  temp = ch(~blacks,:);
  % scale to 0..15 and truncate:
  scale = 15/255;
  temp = floor(temp*scale + 0.5)+1; % 1..16
  tempsize = size(temp);
  for rr = 1:tempsize(1)
    ahist(temp(rr,1),temp(rr,2)) =  ...
    ahist(temp(rr,1),temp(rr,2)) + 1;
  end;
  % Now make volume == 1:
  ahist = ahist/sum(sum(ahist));
  %return(ahist)
% end of myhist2d

%%%%%%%
% histint2d.m
function intersec = histint2d(chhist1,chhist2)
% first, normalize the hist's:
chhist1 = chhist1/sum(sum(chhist1));
chhist2 = chhist2/sum(sum(chhist2));
% hist intersection is sum of min values:
chhist1smaller = (chhist1<chhist2);
minhist = chhist1; % declare
minhist(chhist1smaller) = chhist1(chhist1smaller);
minhist(~chhist1smaller) = chhist2(~chhist1smaller);
intersec = sum(sum(minhist));


%%%%%%%
% myhist3d.m
function ahist = myhist3d(im)
%param is mxn by 3; N is 16, say
  N = 16;
  im = double(im);
  im = 255/max(max(max(im)))*im;
  ahist = zeros(N,N,N) ;
  % scale to 0..15 and truncate:
  scale = 15/255;
  im = floor(im*scale + 0.5)+1; % 1..16
  imsize = size(im);
  for rr = 1:imsize(1)
    ahist(im(rr,1),im(rr,2),im(rr,3)) =  ...
    ahist(im(rr,1),im(rr,2),im(rr,3)) + 1;
  end;
  % Now make volume == 1:
  ahist = ahist/sum(sum(sum(ahist)));
```

```
  %return(ahist)
% end of myhist3d

%%%%%%%
% histint3d.m
function intersec = histint2d(hist3d1,hist3d2)
% first, normalize the hist's:
hist3d1 = hist3d1/sum(sum(sum(hist3d1)));
hist3d2 = hist3d2/sum(sum(sum(hist3d2)));
% hist intersection is sum of min values:
hist3d1smaller = (hist3d1<hist3d2);
minhist = hist3d1; % declare
minhist(hist3d1smaller) = hist3d1(hist3d1smaller);
minhist(~hist3d1smaller) = hist3d2(~hist3d1smaller);
intersec = sum(sum(sum(minhist)));
```

13. Implement an image search engine using low-level image features such as color histogram, color moments, and texture. Construct an image database that contains at least 500 images from at least 10 different categories. Perform retrieval tasks using a single low-level feature as well as a combination of features. Which feature combination gives the best retrieval results, in terms of both precision and recall, for each category of images?

    Answer:
    As a start, use CBIRD, online, for this task. Simple features such as color histogram (or chromaticity histogram) are included in the textbook website, as the Visual Basic project VbColorIndexingSRC.zip under Student Projects > Color Indexing.