48

Display and Presentation Software

Felix Margadant

INTRODUCTION

The final destination of much of the data you coax out of your microscope is the screen in front of a digital projector. Demonstration software such as PowerPoint, Keynote, QuickTime, and MediaPlayer will convert your digital micrographs and movies into a format suitable for projection. What effect does this conversion have on the data you plan to present? We have endeavored to assess what can be properly presented and identify the most notorious pitfalls of two-dimensional (2D) display software.

This chapter supplements information in Chapter 32 about the performance of the display itself. The final result any viewer sees depends both on the display software output and the quality of the projector. Unfortunately, it is impossible to separate the quality of the result from the nature of the images or animations to be displayed. Some image data will lose more quality and look inferior on some displays using some software than others. Therefore, we will first categorize the images into classes. In this chapter, we concentrate on the output the software generates rather than the visible result on the screen. This way we have a logical separation between the rendering of images and the viewer's visual impression.

We will start by considering the display of still images in terms of the resolution reproduced, artifacts, intensity dynamics, and spectral reliability (i.e., color changes introduced). Later, for the dynamic case (i.e., movies), we will assess the frame rate artifacts, frame drops, and format issues. As we shall see, the choice of data format is a much more critical issue for motion pictures than for static images.

In short, this chapter is a list of shortcomings of display tools and display file formats. In contrast to a few years ago, we found no serious flaws in the way that static images were handled by any of the viewing software inspected. The rendering artifacts that at one time were created solely by naive programming are now gone. The pool of cryptic and incoherent graphics formats has shrunk to a manageable number of common formats. And even for the animation formats, Darwinian selection has preferred those that actually work.

As a result, it makes sense to focus on a few encoder techniques and file formats and thereby provide a decent coverage of the fundamentals. Because all the common tools grew from the few surviving concepts, the file formats used actually do work on many computer systems and will probably remain accessible for several years. We cannot cover the myriad software add-ons used to tune the basic display software.

The display errors of particular interest are those made when images are resized to fit into the slide windows of the presentation software. Most of our measurements relate to Microsoft PowerPoint and MediaPlayer, Apple Keynote, and QuickTime.

Although we usually create a new presentation using a computer system with a screen resolution that is likely to be better than 1024×768 (XGA) pixels, projection systems rarely exceed XGA resolution. All the software we investigated kept the presentation freely scalable. When you import a 1024×1024 image of any format, PowerPoint, Windows Viewer, or QuickTime decodes it by means of a software tool specific to the image format and then resizes it by means of the native viewer to fit the window that the presentation uses. If this window or the screen resolution is resized again, then the software re-applies the same scaling mechanism to the first result and this degrades the image in the ways described later in this chapter. None of these viewers modify the original data, which remains unaffected by these tools. However, the image displayed is still the result of the universal decoding mechanism and the universal resizing mechanism we investigate next.

As we shall describe in more detail, the resizing method performs an interpolation between the original pixel intensities. The values are treated as if they linearly evolve between the actual image pixels, and if several pixels have to be downsized into one, their intensities get averaged in such a way that the "old" pixels located closer to the location of the "new" pixel is contribute more. The term used for this method is **smart bilinear resampling**. The claim to be "smart" comes from its ability to stretch an image as well as average it down to a lower resolution. The averaging is crucial to prevent loss of brightness information. It is called bilinear as it works in two dimensions simultaneously. Bilinear resampling is popular as it is very fast. It is also available in hardware, and rescaling introduces almost no distortions to the images. Hence, it is applicable to a vast class of different images such as all photographs. For micrographs, however, better methods are known. Therefore, if you know that resizing will be necessary, it is wise to resample your images before inserting them into the presentation.

The most common error in using presentation software is not using enough pixels. One cannot squeeze all of the information of a 1024×1024 confocal image into a small patch of a 1024×768 PowerPoint model page. Even if the image array does match the screen resolution, it will nevertheless very likely be scaled because the presentation software will try to achieve a perfect match of the geometry (i.e., the shape and aspect ratio of the presentation window) at the expense of matching the image data pixel-for-pixel to the resolution of the screen. Movie players do this too when operated in full-screen mode, and hence it is important to investigate the effects of minor scale changes as well.

These matters lead to a list of precautions:

- Never use under-sampled image material, because it will not properly scale — not even by 5% — and cannot be processed properly.

Felix Margadant • Zurich, Switzerland

Handbook of Biological Confocal Microscopy, Third Edition, edited by James B. Pawley, Springer Science+Business Media, LLC, New York, 2006.

- If you have to use such material, force it into a consistent form. Use a smoothing filter such as a Gaussian with a radius of larger than 0.5 (i.e., 0.7) to make the image consistent for further operations. This will not give you back a proper image but transformations won't further harm that image to the same extent. Deconvolution of confocal and two-photon data, as is recommended elsewhere in this book, will also ensure a properly sampled image.
- Even when correctly sampled, do not scale down the image any more than absolutely necessary. Reducing image size by 50% or more will likely show aliasing artifacts. As the original image noise worsens those effects, try never to scale down by more than 25%.
- If you are really short of space, it is better to crop the image, and cut away the bits that are not absolutely essential, than to simply rescale.
- Scale the contrast so that it looks acceptable in a bright room. If you cannot see the feature of interest on your screen under these conditions, the projection system is unlikely to do better (see Chapter 32, *this volume*).
- When using contrasting colors, keep in mind that your monitor and the projector do not have the same calibration and the latter may not preserve hues. Macintosh users are often surprised when the color consistency of their system does not transpose to the projector.
- If planning to use compression, test the results by zooming in to enlarge details first. If compression artifacts do emerge, lower the compression ratio.

When we show movies, the presentation program does even more work in the background. In particular, because movies are such big files, compression becomes necessary but serious compression requires many computer cycles, impairs the visual quality, and, as we will demonstrate, can slow the presentation.

When you play your presentation, the software renders all the files that go into each slide to make a 1024 × 768 video stream that is passed from the CPU to the graphics board and from there to the monitor or projector.

TESTING

To quantify the impact of display software or imaging algorithms in general, we use a feedback mechanism based on simple artificial images, called *the reference* that are fed to the displaying software (also referred to as the *viewer*) and a screen capturing program (named the *grabber* or *screen capture*) that reads the content of the display memory at a specified time and hence obtains an unfalsified copy of what will actually be displayed. The difference between the reference and the grabbed result allows us to assess the quality. Metrics for the quality are a matter of taste and we use the maximum norm and the Gaussian norm for reasons of economy and broad acceptance; that is, we measure the maximum deviation and the root mean square of the reference, if and only if a norm can be used. If the images cannot be matched geometrically, we only measure the distortions. The reason for not assessing the quality of any resized image is that to do so one would have to decide how to resample the reference, and this would introduce errors specific to the resampling algorithm chosen.

Resampling to change size can either be thought of as changing the pixel dimensions of the display raster or as changing the amount of the original image data that will displayed as a single pixel. The choice of the algorithm used to generate a pleasant-looking new image is driven by speed requirements, and by a fair bit of superstition about the efficacy of different mechanisms of resampling. The dominant paradigm in computer graphics is that a pixel is a point measurement that is sampled at its center coordinate (this is the official notion for the MPEG movie standards), whereas for a microscopic image, this is the light distribution in the specimen, convolved with the point spread function (PSF) and either sampled from the time-data stream from the photomultiplier tube (PMT), or integrated over the area of a charge-coupled device (CCD) pixel. Nyquist sampling implies that the signal from a point object will be spread over 12 to 16 2D pixels at the detector plane (see Fig. 24.A6, *this volume*), and therefore that the signal in each of these pixels represents only part of the image of a point. Because optical systems can be thought of as working in reverse, we turn this situation around to see that, if a single pixel on the CCD is imaged back onto the object, the image of the pixel will cover an area about 12 to 16 pixels in size. This means that the data recorded in each pixel basically represents signal generated from a volume in the object that is the size and shape of the PSF, an area significantly broader than the physical dimension of that pixel. This feature ("fuzzy edges") allows a variety of linear image-processing procedures to be carried out without seriously damaging the image data. However, if the image is downsampled (i.e., compressed into a smaller number of pixels so that the smallest features becomes <3 pixels wide), this relative immunity from harm is lost.

To assess timing errors in motion displays, we used a simple optical probe: a photodiode in a tube mounted to the display (Fig. 48.1). By intentionally encoding specific brightness fluctuations into the test movie (a blinking beacon and a sinusoidal flash) we can observe timing errors. If the computer time source is accurate, however, almost everything observed with the diode could also have been measured with the grabber.

One notable exception relates to the hardware-accelerated movie players that employ modern, built-in *digital rights management* (DRM). DRM basically is a copy protection mechanism for digital movie material and it is designed to prevent movie displays from being captured. As we wish to assess the performance by doing exactly this and as the DRM mechanism is active not only when playing copyrighted material but is also incorporated into some presentation software, it can prevent us from grabbing the output from the test material.

DRM works by taking over some features of the graphics hardware so that the rendered movie never gets copied into the video

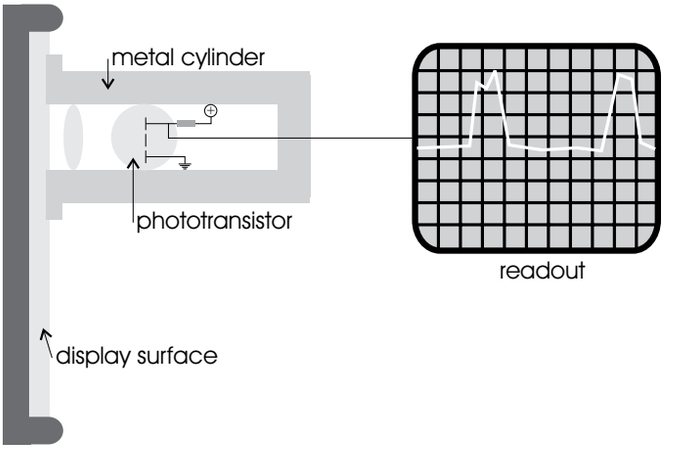

FIGURE 48.1. Test setup for display speed and sensitivity.

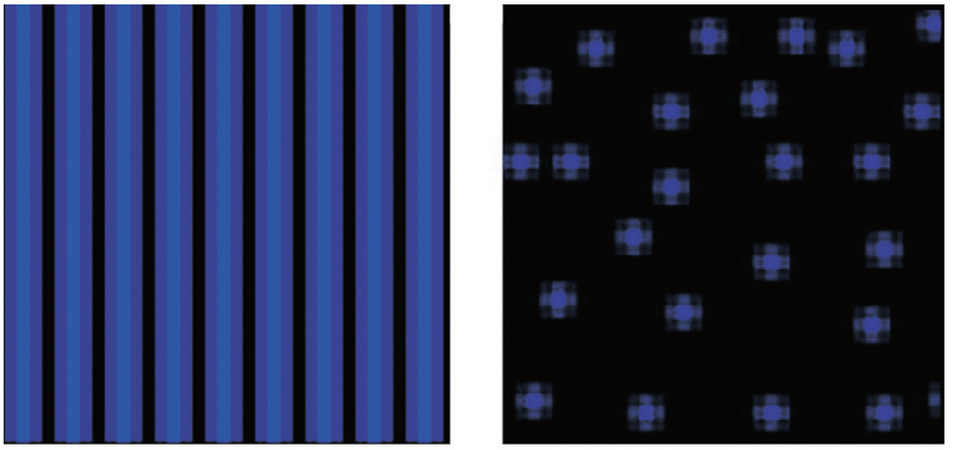

FIGURE 48.2. Sinusoidal grid and sparse dot reference images used to test image reproduction quality.

memory (the method is referred to as overlaying). To gain quantitative insight into the movie playback, we must first disable this hardware feature. However, to assess playback speed and continuity, we must use overlaying, as it improves the rendering performance of the player software, something that we assess using the external timing signal.

During testing, it became obvious that performance limitations usually had less to do with the viewer software than with the underlying universal graphics library that is part of the operating system and hence is shared among all or almost all the graphics programs running on a given computer. If the display module is provided by the operating system, we refer to it as a native viewer in Windows terminology. Furthermore, an image or movie file format is interpreted by dedicated, shared software. This software program is referred to as "codec" (abridged for coder/decoder).

Although these universal software modules keep quality high and independent of the viewer used, they also make it very difficult to assess errors because both the test image generator and the viewer that is being tested likely rely on the same basic software codec.

Codecs are always paired with native viewers: the native image viewer uses a universal codec to handle JPEG images, the native movie player will use codecs for the movie formats used. This layered concept is not new (introduced in 1970), but recently its usage has become far more pervasive. In the end, many presentation packages make use of the underlying graphics subsystem. On a Windows machine, this consists of the Direct X and Direct 3D system; Unix platforms mostly use X Windows; and Macs offer the QuickDraw3D system (see Apple developer site). The graphics subsystem in turn tries to engage the particular hardware features of the particular graphics card that is part of each individual computer, frequently referred to as graphics engine or just the engine. For recent technology of graphics subsystems, see http://developer.nvidia.com/page/home or http://www.ati.com/developer/index.html.

Use of a fast graphics card can have a very positive impact on the rendering speed of the system but it also decouples the performance of the graphics almost entirely from the overall system performance. As a result, we were unable to relate processor power directly to the frame rate of embedded movies. Even worse, there seemed to be no immediate link between the performance of the graphics engine and the movie frame rate, as the entire chain of devices — disk, memory, processor, graphics bus, graphics engine — multi-linearly controls the actual performance.

One result is that the quality of the image displayed on the screen may depend more on the hardware installed on the computer driving the projector and on how this hardware is configured than it does on the program you use. Because Apple specifies minimum quality standards for the graphics engine (http://developer.apple.com/), there seemed to be no (noticeable) difference between software and hardware rendering.

"Static" Image Performance

The setup for testing the ability of the program to display static images is trivial because the resulting image can be grabbed anywhere. The resolution reference test image is a sinusoidal grid at various spatial frequencies and all quantities of interest can be directly extracted from the grabbed image. Another reference used to illustrate under-sampling is an image of sparse dots [Fig. 48.2(B)]. Because it is more difficult to interpret results related to the latter pattern most of the figures below concentrate on the grid [Fig 48.2(A)].

To obtain meaningful results, it is of utmost importance to use properly sampled images with finite line width (i.e., lines at least 4 pixels wide) (Bracewell, 1995), as all modern compression and coding algorithms will fail when applied to images with single-pixel shot noise or steep pixel-to-pixel transitions as we shall illustrate.

The native viewer for Windows XP is the Windows and Fax Viewer; for OS X it is QuickTime. The first finding is that, not surprisingly, when image data is displayed without data reduction, the native viewer does not change the image intensity content at all as long as the image has exactly the same raster size as the graphic window in the slide. Any image with three or fewer channels and 8 or less bits per channel, is displayed as is. This also means that the output of different viewers is indistinguishable, and as all PowerPoint versions later than the 1997 edition use the same underlying graphics library for viewing static images, images that have not been scaled in any way are reproduced accurately; that is, the image grabbed at the output of the computer is identical to the source.

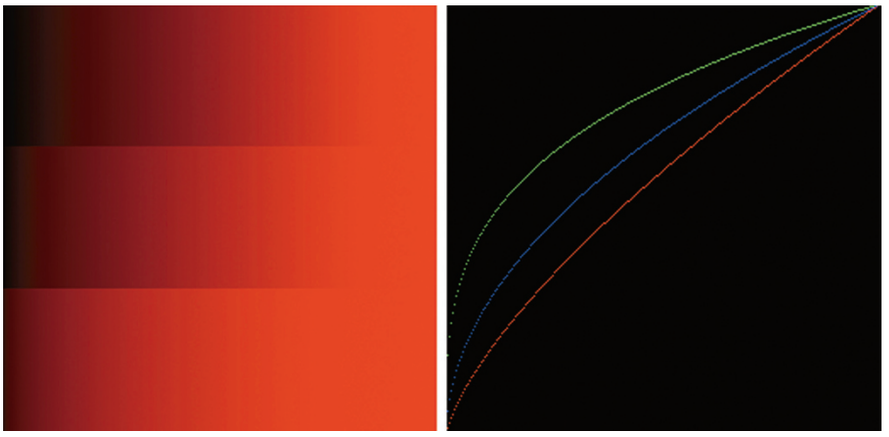

FIGURE 48.3. Strong photometric gamma switched off (left) and on (right) for a linear ramp. The visual brightness is red; the absolute value plotted in green.

Brightness

This equality does not extend to the brightness correction information that is stored with each image. Such information is a feature supported by both the JPEG and TIFF formats.[1] This photometric information (also termed **photometric interpolation**) is intended to standardize the actual brightness of a digital value stored with the image. This is not done as an absolute physical value — which would be difficult to display and print — but as a ratio to the peak brightness displayed. Hence, all packages make adventurous assumptions about the dynamic range and linearity between the darkest and the brightest values displayed. These assumptions range from assuming that everything is fine (i.e., completely ignoring the photometric information as the Windows native viewer does) to properly scaling the image to the photometric ramp (as in Photoshop). When the display contains a stored profile describing its color response, these editing tools will scale the output to take this profile into account.

For example, when Windows 2000 is used with modern graphics cards, the Direct X library includes built-in correction for the display. This has always been an intrinsic part of the system for Apple machines and hence is never an issue.

The Nvidia and ATI graphics drivers refer to this feature as monitor gamma, gamma being the inverse of the brightness response of the display. Fortunately for the user, these values are transparent and can be read by the software. If a monitor contains a gamma specification file, Photoshop and Paintshop image editors will use these preset display values and not add their own correction system. In the end, the results can vary considerably if this feature is used (see Fig. 48.3).

For the user, all of this hidden activity has a very limited impact. Micrographs represent a quantity related to photon counts and hence do not carry any photometric correction information. There is no photometric meaning to microscopic images other than that the brightness scale should be as linear as possible and this is close to the case where no photometric scaling is used.[2]

Conclusion: do not actively adjust the photometric information until one has finalized the image editing for printing. To do otherwise may contaminate your results without adding anything to your image quality. Faulty calibration ramps (or ramps that are considered faulty by an editing program) can sometimes reset your rich, beautiful image to black.

Once the layout is ready for publishing, any photometric information available assures that the results stay constant as long as the images are used with the same set of programs (i.e., an image saved in Photoshop will look much the same on different displays with known profiles). To test this claim, we started with an image that was the inverse of the gamma ramp shown in Figure 48.3(B) and processed it through the same gamma-correction photometric ramp so that the output should mimic the linear ramp. The original data is shown in Figure 48.4 (top) and the reconstructed ramp is seen at the bottom. It looks as expected, except that, because the computation is done in integer numbers, the ramp looks coarser because of round-off errors The colored bars mark lines of identical brightness. These marks are shifted but the effect is quite mild and hard to spot by eye.

Resolution: Changing the Display Size of Your Images

Desktop and notebook displays are now available at decent resolutions. XGA (1024 × 768 pixels) represents the very low end, SXGA (1280 × 1024), SWXGA (1400 × 1050), UXGA (1600 × 1200), UWXGA (1900 × 1200), and Apple's 1200 × 800, 1280 × 854, and 1440 × 900 can be found in low- to medium-priced devices. Desktop screens come in these and higher resolutions. Projection systems, however, feature SVGA (800 × 600) or, more commonly, XGA (see Chapter 32, *this volume*). Higher resolutions are still rare even though SXGA projectors do exist. Briefly: a beautiful micrograph that is 1024 square will simply not fit on today's projector. PowerPoint and the native viewers "solve" this issue by down-sampling the image if needed, using the graphics library and the graphics hardware. Using the test setup to compare different resampling algorithms (Nikolaidis and Pitas, 2000), it is obvious that presentation software uses bi-linear interpolation, as illustrated in Figure 48.5 It is a fast resampling method with the only disadvantage that it does not correspond to any physical system, that is, it cannot be simulated by building an optical system that changes the resolution in the same way.

Although the inherent and unavoidable danger of all down-sampling is the loss in resolution, a fine feature, correctly sampled at the resolution limit, will no longer be correctly sampled when

[1] http://www.jpeg.org/ and http://www.digitalpreservation.gov/formats/fdd/fdd000022.shtml.

[2] The linearity (or not!) of both the display and the visual system is discussed at greater length in Chapters 4 and 32.

FIGURE 48.4. A linear ramp and a gamma ramp with inverse correction.

scaled down. This can be visualized by observing the effect of the process on the line grid image that originally was properly sampled.

To illustrate the damage done by down-sampling in PowerPoint — or linear interpolation in general — we force a 25%, 50%, and 75% scaling of the line grid reference image in Figure 48.6.

Reducing the resolution of the line grid by 25% to 50% hurts the resolution but it does not totally destroy the image content as the undulation between high and low persists through the scaling even though the peak values of the resulting images are less than they were in the reference. This is because, while the 50% scaled image is no longer properly sampled, it is not yet *aliased*: it is the smallest scale at which all details of the original image will still be present. When new structures, such as patterns, appear during the manipulation of a raw image, the phenomenon is referred to as aliasing. When a Nyquist-sampled image is down-sampled to 25% of the original size, it will suffer from severe aliasing even if the down-sampling is performed correctly through the interpolation. As the intensity of four neighboring pixels is averaged, the image becomes homogeneous. This (correct) algorithm for reducing image content to fit it into a smaller window is built into the modern libraries; however, on systems using older software, the interpolation can still go wrong, as Figure 48.7(D) shows.

The ability to sample over more than just the nearest-neighboring pixels when scaling down is called anisotropy (http://www.opengl.org). Because the fast preview modes of Photoshop or Paintshop are often used without anisotropy to increase speed, this phenomenon can still be observed, but it is not an issue for presentation software.

All these cases were "well behaved" in the sense that down-sampling them did not introduce any structure but merely lowered the image quality. However, even with properly sampled images, it is easy to commit true mistakes. Figure 48.6(B) illustrates the common failures of digital image operations. Scaled to 40% of its original resolution, the image content breaks down regardless of the interpolation method chosen. Even though the down-sampling was done correctly, the original structure simply cannot be displayed anymore.

Although the 50% image still conveys the original information [Fig. 48.6(C)], it cannot be manipulated any further. Any rotation or shifting more complex than a 90° rotation or shifting an integer number of pixels will then destroy content, as Figure 48.7(B) proves.

Far stronger under-sampling pattern artifacts (for regular structures, these are called moiré patterns) can occur, as Figure 48.7(C) shows for the 30% scale image. Instead of displaying fine lines, the image exhibits slow undulations. Although similar errors occur from non-regular patterns — such as micrographs — they are harder to recognize. Except when the scanning process itself creates some regular pattern, excessive down-sampling is more likely to break up structures or attach them to neighbors than to create a large-scale pattern (see Fig. 48.8).

How can we prevent such things from going wrong? As XGA can safely be regarded as the low-end display for computers and is the dominant display for projection systems, if scaling your Nyquist-sampled original image down to an XGA display does not force you below the 50% threshold, then details should remain visible. So a 1024 × 1024 Nyquist-sampled micrograph can be

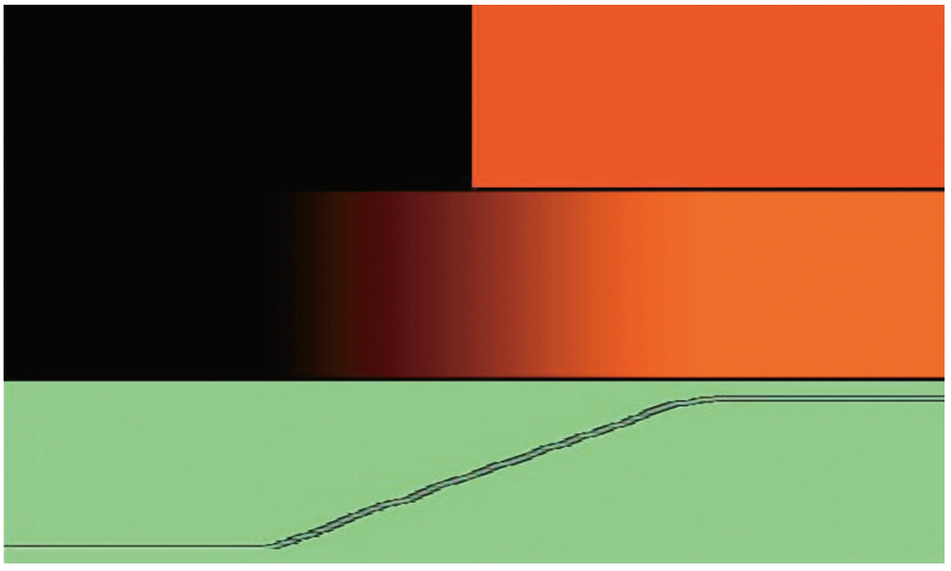

FIGURE 48.5. When a step image (top) is zoomed by PowerPoint, it results in a ramp (center). The intensity plot (bottom) shows a ramp consisting of three linear segments.

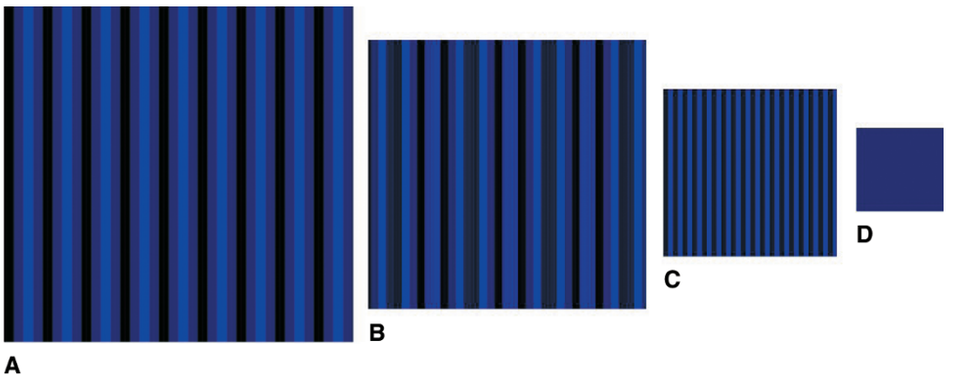

FIGURE 48.6. Down-sampling in PowerPoint. From the left: original image, 25% smaller, 50% smaller, and 75% smaller.

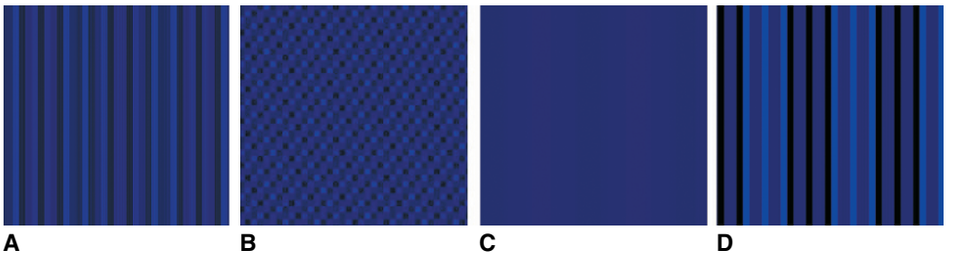

FIGURE 48.7. Classical aliasing gallery: (A) The reference image sampled at 40% of its original resolution; (B) the 50% size image, rotated by 30°; (C) the reference image reduced to 28% of its size; (D) the image at 70% if its resolution with nearest-neighbor interpolation instead of linear interpolation.

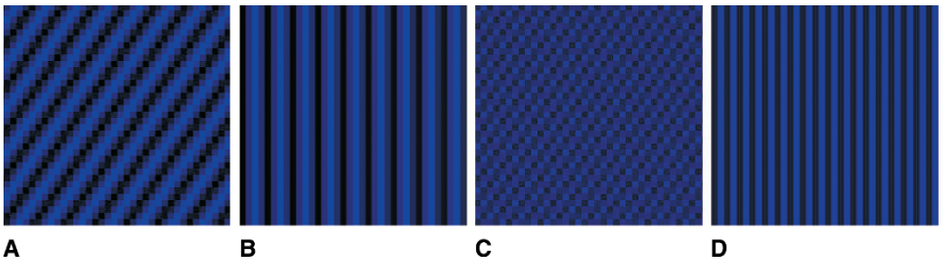

FIGURE 48.8. Operations that do not produce aliasing in properly sampled image data. If you subject your beautiful micrograph to any of these operations, keep the worst case in mind and do not allow it to be scaled down by more than a factor of $\sqrt{2}$.

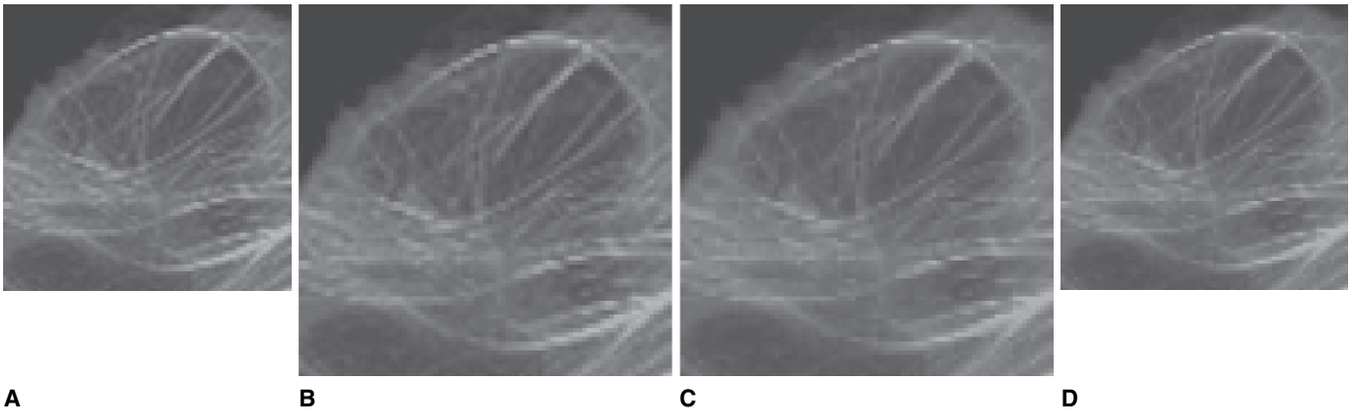

FIGURE 48.9. Up-sampling: (A) a slightly under-sampled image gets up-sampled by 30% with either full anisotropy (B) or basic linear interpolation (C), as used by PowerPoint. For comparison, the pixel resolution in (D) is reduced by 30% to see the similarity of the processes.

shown in a presentation as long as it occupies at least two thirds of the display's height. However, you can only put two such (now 512×512) images side to side for comparison if you utilize the entire width of the screen. If you want borders, you must *crop* the images to cut away all but a *region of interest* (ROI). This is often a good idea anyway as those at the back of the hall can usually not see the finest details on the screen.

The process of scaling, rotating, removing distortion, and aligning images all lower the effective resolution of the digital image, but the order can be important. If you shrink your image into fewer pixels, first, and then rotate or distort it, you will now be working on an under-sampled image and the damage will be more severe: Shrink last!

Digital images have a $\sqrt{2}\times$ coarser resolution in the diagonal direction than along the axes because of the rectangular grid, a property not shared with analog media. If an image is turned by an angle $-45° \le \alpha \le 45°$ away from the axis, the values along the grid will be mapped onto a raster of $\cos(\alpha)$ times the resolution with the $1/\sqrt{2}$ worst case for 45°.

Translating a digital image a fractional number of pixels can also blur the image. Fortunately, the effect is less pronounced on properly sampled data [Fig. 48.8(D)] than is the effect of scaling and rotation [Fig. 48.7(A,B)], but the resulting images lose resolution, as the half-sized images of the rotated reference [Fig. 48.8(C)] and the shifted image [Fig. 48.8(D)] show. The latter shows no loss at all but the shift is no longer noticeable, except for a mild change in intensities.

Even if the damage done in Figure 48.8(C) appears minor, you need to remember that here the operation was done in a single step with maximum care and had the same result been reached as the result of a number of separate steps, errors would have accumulated at each step (Fig. 48.9).

Other types of operations, such as changes in histogram, linear filters, convolution, and deconvolution, do not impair the resolution and little care has to be taken when using them. Median filters, anisotropic filters, and image editing **do** harm image resolution and, as the images may no longer be properly sampled after these steps, caution has to be applied. Even minor changes of a page setup — without any intention to rescale a picture[3] — can result in changing the resolution of the image.

Compared to down-sampling, up-sampling is relatively harmless. Therefore, the safe approach is to keep the image near the target system's resolution and assume that small changes in scale will not harm the content.

Figure 48.10 shows that this happy outcome cannot be fully achieved if the image itself is not properly sampled. This means that even if we can make sure that the images never get reduced in size during a presentation, proper sampling of the original image is still essential to obtain a consistent outcome.

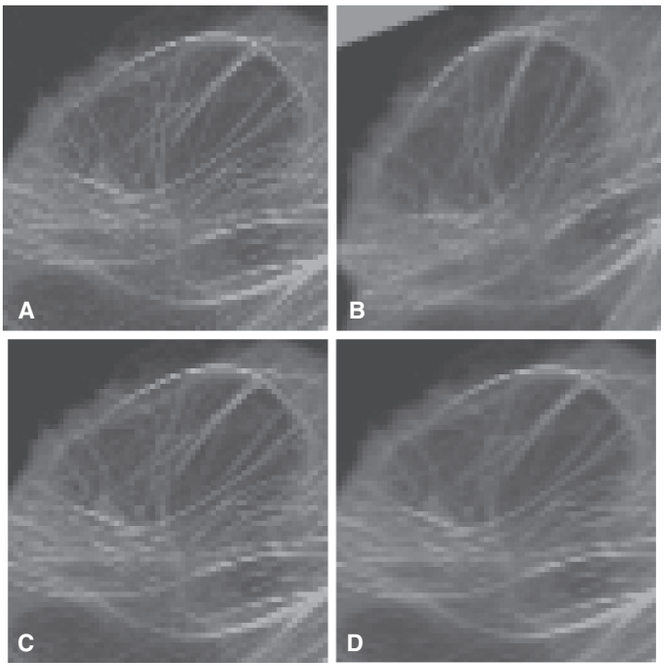

FIGURE 48.10. Manipulation of the slightly under-sampled image. From the original (A), a 30° rotation (B) and a half-pixel shift were performed with sinc interpolation (C) and with linear interpolation (D). The artifacts arising are least pronounced in (C), but all result in some loss of resolution.

Compression

Although compression is often unnecessary for still image presentations, it also has some significance for storing large datasets and hence we briefly touch it here. Classically, compression algorithms are split into two groups. The first type are pure compression methods (so-called no loss methods) that try to exploit redundancies such as homogeneous regions in the image to use less information to encode the image. The second type are data-reduction algorithms that try to identify and remove image information that is not needed for conveying the essential information. As such a compressor cannot really know what it is that you want in an image, using such a procedure is a common source of artifacts.

As light microscope images are inherently noisy and noise has no redundancy, it is important to ALWAYS deconvolve[4] your 3D confocal data before trying to display it. Doing so not only suppresses single-pixel noise, but also effectively averages the signal from many voxels to reduce Poisson noise. Noisy images cannot be compressed noticeably by means of pure compressors (Castleman, 1995). Exceptions to this rule are images with many "black" pixels, as dark regions exhibit little noise. It is trivial to test if your images actually can be compressed. Start with a format such as TIFF that offers pure compressors to squeeze the image and then compare the size of compressed image to the original.

What is lost when compressing microscope images using lossy methods? All modern compressors rely on psycho-visual perception compression (Watkinson, 2001), that is, they try to obliterate any image information that the human eye is bad at reliably picking up. The most famous such reduction is the "spectral coding" which relies on the observation that the spatial resolution of the human eye for color changes is much lower than for intensity changes. How well the compressors take advantage of this fact is shown in Figure 48.11.

Most compressors exploit the connectivity of neighboring pixels. If the image is noisy or not properly sampled and hence does not exhibit this connectivity, the effort to compress will either fail to achieve high compression or — much more likely — introduce errors to the image. It will also often NOT produce a smaller image file.

How does compression affect the display of micrographs that not only should look pretty but must also convey correct bright-

[3] For example, changing the printer reproduction ratio.

[4] Or at least Gaussian filter for 2D data.

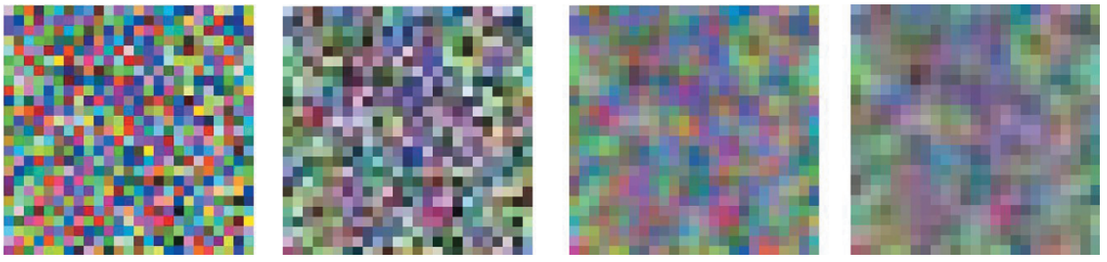

FIGURE 48.11. A random color dot image (A) gets mildly compressed (B). Although the overall hue is preserved (for a display of unity gamma), the individual colors get completely washed out. If, however, the original image is properly lowpass filtered to remove the single-pixel noise (C), the compressed image (D) looks very comparable.

ness information? We focus on a small 200 pixel square subsection of a properly sampled micrograph (with some noise) and assess what compression is doing (Fig. 48.12). First, because the image contains a large black background with relatively low noise, the image can be compressed with little loss by a factor of 3.97.[5]

Even though we get very small error values even for the strongest compression, the visual result is poor as the fine details are lost and this is what we perceive as quality. For image processing, this has substantial consequences: even mildly compressed images are no longer suitable for deconvolution, segmentation, or quantification. However, for presentations, compressed versions of properly sampled images will usually leave a good visual impression.

MOTION PICTURES

Athough motion pictures inherit all the artifacts that plague still pictures, the image changes so fast that the flaws are usually less noticeable. On the other hand, they add timing flaws. The dominant problems are non-steadiness in the picture flow (jittering) and dropped frames because of lack of computer power. In addition, there are implementation mistakes that can create incorrect frame rates or random intervals between frames. The latter are easy to detect and usually require the movie file to be re-created.

As for image-viewer programs, the movie player software is only responsible for a fraction of the possible problems. Once a movie is created, certain flaws can no longer be compensated for by the player software. Because movies that exhibit aliasing in resolution or timing cannot be resurrected by better player tools, correctly coding the movie will be our first focus. The second case of interest is when player and data are set up correctly but do not match the hardware systems they run on.

Movies also add another entire dimension to the problem: that of computing resources. Movies use large amounts of both storage and computing power, and playing a movie at the required frame rate can push a computer over its limits. Also, unless the player software is dedicated to playback-from-disk mode, discontinuities in playback are likely if the entire movie cannot be held in computer memory. These considerations restrict the number of formats that can be used for playing very long time series.

[5] Such compression is possible, at least in part, because all the "black pixels" can be specified as black using fewer bytes than recording a row of zeros. However, the process still entails a "little loss" because setting the "almost black" pixels to zero is a small loss. If the data has been 3D deconvolved before being compressed, few pixels will actually be zero, and one must set a threshold, a process that may also entail some small loss of data.

Of the plethora of animation formats available, we investigate only the most popular ones. There are also the *metaformats* QuickTime and Microsoft's AVI that are really just descriptions of the parameters of the movie such as frame-rate, resolution, color matching, etc., but that do not decode the movie data itself. Within these formats, there is also a parameter that refers to the codec to be used to interpret the movie data. Hence, these metaformats can be reduced to the understanding of the codecs involved. To keep things nice and complex, the word QuickTime is used both as the name of a codec and also as the name of a metaformat. Below, we talk about the basic problems of the codecs rather than the convenience of the player software (i.e., MediaPlayer and QuickTime player).

The codecs investigated are QuickTime, Cinepac, MPEG 1, 2, and 4 (Ebrahimi and Pereira, 2002) [including the new MPEG-4 offspring, H.264 (Richardson, 2003)], and the image-series format. The last mentioned features no movie compression at all but simply consists of a series of frames. The frames themselves can still be compressed but there is no link between the frames other than enumeration. The format comes in a few popular forms, including TIFF series (which simply uses the TIFF image standard for frame handling), Motion-JPEG that uses the JPEG standard (which is incorporated in the mini-DV, digital camcorder format) and the Digital Betacam (broadcasting) formats. These compression standards will now be discussed in more detail.

As we shall see, the following considerations have to be respected:

- Shrinking movies results in the same artifacts as shrinking stills. If scaling down is required, it is wiser to shrink the frames by proper down-sampling before they are encoded into the movie format. However, cropping movie formats should also be considered (Fig. 48.13).
- Movies can also be under-sampled temporally. Depending on the recording device, this can be much harder to prevent than spatial under-sampling. It becomes obvious when structures move enough between frames that they cannot be properly associated in the individual frames. Our vision system compensates a substantial amount of temporal aliasing by utilizing other criteria such as shape, hue, and texture to label moving objects, but this skill is limited to a few objects per scene. Temporal aliasing can be reduced either by increasing the recording frame rate or by reducing the complexity of the scene with proper color staining or labeling.
- If none of these are possible, the best remedy is to play the animation at a very low frame rate so that the visual cortex gains time to understand the scene in each frame. The result resembles a slide show more than an animation.

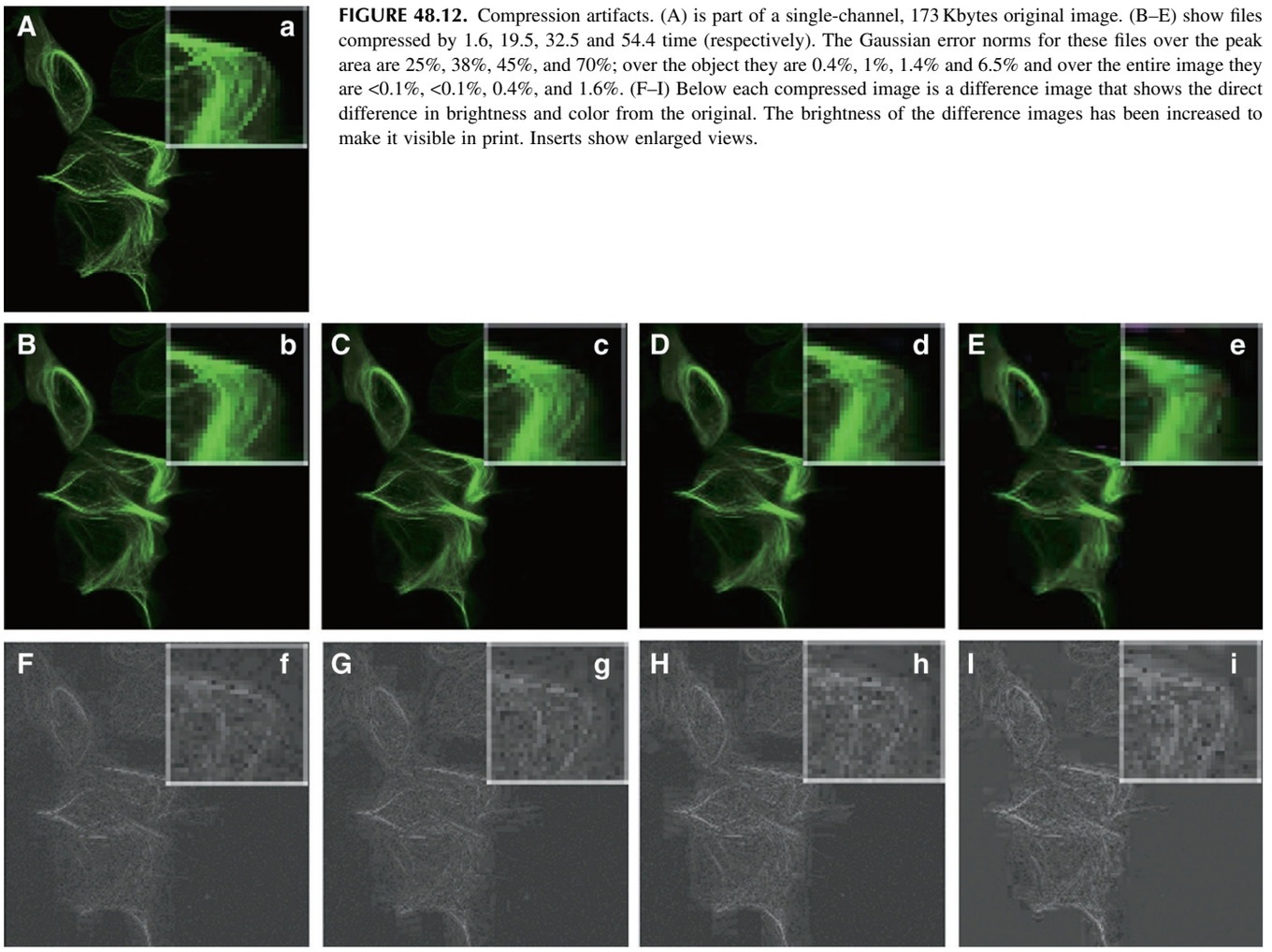

FIGURE 48.12. Compression artifacts. (A) is part of a single-channel, 173 Kbytes original image. (B–E) show files compressed by 1.6, 19.5, 32.5 and 54.4 time (respectively). The Gaussian error norms for these files over the peak area are 25%, 38%, 45%, and 70%; over the object they are 0.4%, 1%, 1.4% and 6.5% and over the entire image they are <0.1%, <0.1%, 0.4%, and 1.6%. (F–I) Below each compressed image is a difference image that shows the direct difference in brightness and color from the original. The brightness of the difference images has been increased to make it visible in print. Inserts show enlarged views.

- The temporal/spatial aliasing dualism of movies also works in the opposite direction. If a movie is sampled properly in time, it conveys an impression of having higher spatial resolution than a still image of the same pixel dimension. This is obvious if you digitize the TV signal. A single frame will appear much coarser than the equivalent animated picture. This means that for movies at high frame rates, the resolution can actually be lowered without seriously reducing the apparent image quality.
- There are only a few universal movie formats. If you stick with the widespread MPEG family of animations, frame rates and resolutions are specified and your movie will have to be trimmed to fit both. The reward for doing so is that you obtain a document that will play reliably on almost all platforms.
- If it is compressed, movie performance depends on the complexity of the movie, as both decoding effort and memory requirements increase with complexity. The exceptions are the MPEG-1 and MPEG-2 formats that offer complexity-independent performance.
- Movies may easily exceed the memory resources of the computer and even when they don't, they may take long to time load. As a result, it can be hard to tell whether the movie is just slow in loading or if it will not play at all because the presentation machine is not sufficiently equipped. The MPEG and QuickTime formats start playing after loading only a small part of the presentation, but only MPEG guarantees proper playback performance if played from the disk.

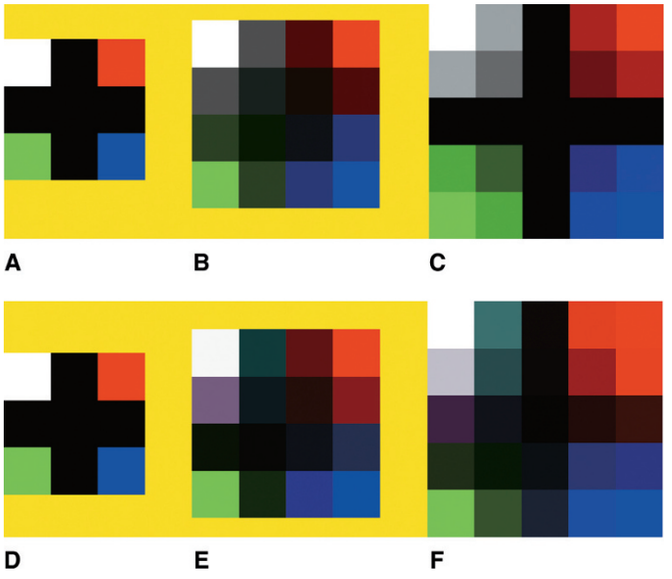

FIGURE 48.13. A 3 × 3 color sample is zoomed to 4 × 4 and 5 × 5 pixels. Linear interpolation (A–C) as it is offered by movie playing hardware versus high order interpolation (D–F). The main advantage of high order interpolation is that the borders stay sharper, even though the image looks no more convincing on such a strongly zoomed scale. (Try looking at it from across the room.)

As with static images, the output of motion players can be grabbed by software grabbers unless the player has the rights-management software installed (as mentioned above). The grabbing setup is synchronized to the refresh rate of the display and during each displayed frame, a small, constant region of the screen is captured as a static image that then can then be compared to the content of the original movie file. To allow for exact matching, frame numbers were written into the test movie file. These can be read by the grabbing software, permitting a one-to-one comparison to reveal when a frame is displayed or not displayed because it was dropped. This simple setup allows analysis of rendering artifacts, jittering, and frame drops all at once. It is important to recognize that the frame rate must not be higher than the refresh rate of the display. This is not entirely an academic problem, as NTSC videocams can record at 60 Hz and some LC displays, especially the very high resolution ones, are limited to a lower refresh. In addition, many projectors use 30 Hz displays, assuming that this is sufficient for rendering full-frame PAL or NTSC at 25 Hz and 30 Hz, respectively.

It is expected that the frame-by-frame (2D) quality of movie players will be worse than in viewers, because movie players rely on fast codecs, supported by the graphics hardware. There is only a problem if the movie is scaled down during the presentation. As a result, it is even more important to avoid downsizing movies than it is for stills. Besides the speed penalties, shrinking movies can result in coding and playback artifacts, some of which we show below.

Coding Limitations

It is trivial but crucial to keep in mind that the same fundamental restrictions hold for movies as for stills. You cannot safely display a movie with a higher format resolution than your display window. All the players are armed with down-sampling algorithms that — due to the strict timing requirements — are texture based (resorting to the Direct X or QuickDraw library) and use linear interpolation. The price is high: uncontrollable loss of image detail even at constant (or increased, for the sake of down-sampling) computing power. Both QuickDraw and DirectX use the texture hardware to zoom images up and down and this will result in sampling errors, if the hardware anti-aliasing has been switched off. Figure 48.6(D) shows a 4-pixel sinusoidal pattern that has been down-sampled by a factor of 4. Although the proper outcome for this pattern is a homogeneous average intensity, the hardware sampling on the Windows machine used is set to only handle interpolation between neighboring pixels and, hence, the result ends up as shown. To show the same outcome on Apple hardware, the down-sizing has to be more severe, exceeding a shrink factor of 4. Although this might happen if many proper 512 square movies are laid out on a single PowerPoint or Keynote slide, it seems rather unlikely.

Most movie formats do not cover really high resolutions properly so users are saved from that mistake by system restrictions.[6] The only way to keep control over your movie details is to limit its resolution to the worst display you are likely to use.

Apart from resolution, movies face a frame rate limit. Most projectors are capable of handling full frames at full resolution at 30 to 60 Hz, except for the very high resolution models. Cathode ray tubes (CRTs) and liquid-crystal displays (LCDs) offer even higher refresh rates, likewise high-speed cameras. However, as human vision cannot resolve these higher frame rates, it is safe to down-sample them to 60 Hz if real time is required or use slow motion if every frame counts. It should be obvious that, when building a movie, one must respect the computational power of the display system. Not doing so results in jerky image motion and the unpredictable behavior of a player running at an arbitrary frame rate and trying to reduce image contents using the fastest (i.e., lowest quality) method available.

Up-Sampling or Frame Rate Matching

Assuming now that we have a healthy movie file designed to run at frame rate f with a horizontal resolution X that the display can accommodate. We then face a temporal scenario that is similar to the spatial scenario we encountered for stills: the display not only exhibits a constant resolution, its frame rate is also fixed. Although both can be changed, doing so requires setup effort and cannot be done on the fly. Given a fixed refresh rate, the player software behaves in the most canonical way possible: that is, at a 60 Hz refresh, frames start ~16.67 ms apart. On the other hand, an 8 Hz movie that issues a new frame every 125 ms produces a display table that looks like Table 48.1, and this is what one obtains from a sequence of grabbed images — each 7 to 8 cycles, the frame counter increases.

Although the player and movie files are both perfectly correct, the images of the movie cannot be displayed for 125 ms each as designed. The first picture (no. 0) is displayed for 133 ms; the next one gets swapped at 250 ms and hence stays for 117 ms, the third one for 133 ms, etc.

In many cases, the 16 ms difference in display time will be hardly noticeable, a feature exploited by low-end conversion between TV standards, where 25 Hz get mapped onto 30 Hz or vice versa.

If however, you encode two labels in alternating frames and display them in red and blue, then one of the channels will appear $\frac{133\,\text{ms}}{117\,\text{ms}} \approx 1.137$ almost 14% brighter than the other one. Not a disaster, but definitely not what one intended to convey. Worse still, each time the sequence is played, randomly, either the red or the blue feature will appear brighter.

There are two lessons to be learned from this: Do not use the compressed movie formats for anything but their designed purpose. If you need to add stereo or highlighting effects, either the movie format has to support them, or you have to code them into the frames, and cannot rely on the interframe time lapse as it is unstable.

A more important message is to only use frame rates that are literally in-sync with your display. So for a 60 Hz display, use any

TABLE 48.1. Synchronization Between Frame Count and Display Cycles

Time	0 ms	16.7 ms	33.3 ms	50 ms	66.7 ms	83.3 ms
Refresh	0	1	2	3	4	5
Frame	0	0	0	0	0	0
. . .						
Time	100 ms	116.7 ms	133 ms	150 ms	166.7 ms	183.3 ms
Frame	0	0	1	1	1	1

[6] However, this situation is in great flux, as Apple recently moved up its H.264 codes to cover the two MPEG high levels, these formats can now actually be created in a high-definition standard. Of course, they cannot yet be played on common consumer hardware or XGA projectors.

integer fraction of 60 Hz, such as 6 Hz or 10 Hz. Because, even in countries with 50 Hz TV standards, projection systems support and prefer 60 Hz modes when connected to a computer, it is much easier to trim your movie for a digital presentation than for your home television. As some CRTs use high and exotic refresh rates such as 85 Hz, 92 Hz, or 120 Hz, it is futile to trim a movie to their needs.

If you cannot match the refresh rate of the display, you can compute the variance of the frame display time: with t_R the refresh time (i.e., the inverse of the refresh rate) and $t_F \geq t_R$ the frame display time and the refresh cycles per frame q: = t_F/t_R, the time mismatch will be $t_M = (q - \lceil q \rceil)t_R$ (Eq. 1). If $t_M > 0$, then the maximum timing error is $\lceil t_R/t_M \rceil \cdot t_M$ (Eq. 2). From Table 48.1, one would expect that the theoretical maximum error would be near $2 \cdot t_R$ and that this would be worse at the start and the end of a movie clip. However, when we captured a series of out-of-register frames it became clear that the movie player had assigned each frame to its closest refresh period rather than to the next one in sequence, that is, they use a *centric* update that reduces the worst frame timing error by half.

Despite this smart feature, Equation 1 has two noteworthy consequences. If there is any timing error whatsoever, the maximum error will always be larger than half a refresh cycle. Even more unexpectedly, any apparently rational ratio between frame rate and refresh rate will result in a largest possible timing error of one entire refresh cycle! So it is not wise to somewhat align the frame rate; they need to be identical.

Practically, this means that if you cannot align your movie frame rate to the refresh rate of the display (something that you might not know) then a timing error of up to one refresh cycle is nearly unavoidable. This timing error is less dramatic if the refresh rate is significantly higher than the movie frame rate. A 7 Hz movie will look OK on all types of displays, whereas the 50 Hz animation noticeably varies in playback speed when viewed on a 60 Hz device (often referred to as *breathing*).

We keep hammering this issue because, very often, one has a choice as to how fast your animation plays back, and slowing it down might actually improve the visual impression. Also, some advanced movie formats (such as MPEG-2 and MPEG-4) do not encode individual movie frames but evolve one frame from the previous one. These formats can produce frames at any desired point in time, not just at the original frame rate. This feature is called *pull down* and is used for standards conversion between television formats. In the case of MPEG-2, it actually can be performed in real time on the very fastest single-CPU machines available. It does not yet offer the quality of offline pull down but will definitely become a method of choice in the future. The MPEG-2 player from Philips already offers this feature. So far no MPEG-4 player can do online pull down. As an intermediate solution, one could create the same movie for each of the refresh rates one is likely to face, one for your native display speed and one for 60 Hz projectors.

Motion Picture Artifacts

Besides the patch and pixelation artifacts inherited from still images, movies face timing subsampling artifacts (temporal artifacts). If a scene evolves too much from one frame to another, the movie will be plagued with temporal aliasing. As in the case of an image with insufficient resolution, this has absolutely nothing to do with the encoding of the movie or technical restrictions and depends solely on the mismatch between movie acquisition and human perception. Fortunately, because human vision is outstandingly tolerant with temporal artifacts and can compensate for it, temporal aliasing catches much less attention than subsampling images spatially. In addition, automated software that can be applied to time sequences is still much less widespread than image-processing algorithms for still images. Finally, because movie formats are seldom used for image-processing applications, we have limited the issue of temporal artifacts to coding errors.

A movie with a fixed data rate cannot always perfectly encode the amount of change between two frames. The different formats cope with this problem in very different ways. The MPEG formats have switches to handle this scenario. MPEG-4 can be set to "scene change detection" that will cue restarting encoding at steep scene changes. MPEG-2 can be allowed to arbitrarily increase its data rate (called *variable bitrate encoding*) when a scene change requires this. Both solutions come at the expense of higher storage requirements. The other movie compressors also invest more in coding data to handle changes but they do so much less consciously, to the extent that image noise can drive their coding effort to the same level as real scene changes. Conversely, limiting their noise encoding restricts their scene update accuracy.

Naturally, none of this holds if one uses a coding format that does not include any compression. TIFF series are lossless as they offer no data reduction either within the frame or over a sequence of frames. For today's large micrographs of one or more channels and megapixel resolution per frame, these formats have little relevance except for programs that allow us to pan and zoom small windows in these datasets.

The second type of artifact comes not from the information encoded in the file but from insufficient machine performance for decoding the animation or from the limited capabilities of the decoder. This can be a very serious restriction when playing back large frames at high rates, especially if one must use an unknown computer. All MPEG-4 formats, independent of resolution, are computationally expensive. Do not expect an aged computer to perform seamless MPEG-4 playback.

At rigid timing, the player will drop frames, causing motion under-sampling to become even worse. In addition, when forced to play all frames, the player may not decode all the image information and may play coarse images with ripples in them. The results look very similar to badly encoded movie formats in general.

As outlined above, decoders usually do an excellent job of squeezing the best possible result from the movie format and computer resources available.

The same cannot be said about encoders (also referred to as compressors) because the task of the compressor is infinitely complex and as long as the compressors do not understand what they are encoding, they have to rely on heuristics to decide what image information they can throw away without visible harm to the movie quality. After all, this is the only way to reduce the data.

The criteria for useless information are unique to each compressor but all are optimized to handle cinema movies or TV shows, a condition that sets clear limits on the color space they must handle and how well they replay motion. Unfortunately, cinema-optimized encoders are notoriously bad at displaying small moving objects against a large constant background, that is, images that look like a fluorescent object on a black background. This is why some encoders have a special setting to encode sports shows, a setting that can be useful to process time sequences of micrographs. Without this optimization, small moving objects drag strong ripple artifacts around with them. On the other hand, fast motion compensation comes at the price of a generally higher noise level. MPEG-2 encoders can achieve both low noise and

small object motion at once but only by requiring higher data throughput. It is usually worthwhile pay the price of a large data file as it usually costs less than more computing power.

The MPEG Formats

A notable feature of movie formats is that the more advanced formats can arbitrate between performance and quality, and are able to sacrifice resolution and sharpness, in order to maintain the frame rate. Depending on the data to be displayed, this can be either useful or noxious. Two major approaches exist to implement this so-called *rigid timing*. MPEG-1 (as it is implemented in the Video For Windows format, also known as H.261) and MPEG-2 (also named H.262) limit the complexity of the movie format (also referred to as *bitrate* capping) upon coding the movie. If content changes too much between frames, capping can only be achieved by jettisoning some of the resolution. MPEG-2 can encode two different bitrates at once, called the *basic* and the *helper* stream. If performance allows for it, the helper stream is decoded as well and this helps to improve the quality. If the resources are insufficient, the basic stream supplies the output. The concept is not very flexible and it can only either boost resolution or lower noise. The unique feature of MPEG-2 is that because the helper stream neither has to be decoded nor read, playback performance is assured.

Because of this, frame-rate tests for MPEG-1 and MPEG-2 yield trivial results — independent of the scene complexity, the frame rate stays constant and frame dropping will only occur when CPU power is very limited and any sort of movie playback is not really possible anyway. On current low-end 2 GHz systems, the CPU load is below 25% of the CPU resources even for very complex movies (see description of entropy) and hence not worth investigating.

MPEG-4 incorporates MPEG-2 modes but adds the capability of limiting the format complexity during playback. Therefore, an MPEG-4 file can be generated at a high bitrate and then be played back at a much lower one (Ebrahimi and Pereira, 2002). In contrast to the MPEG-2 helper stream, however, all data must be read to decode an MPEG-4 stream and this can impede playback performance. Consequently, the player software has a substantial influence on MPEG-4 performance (Walsh and Bourges-Sévenier, 2002).

Many custom players such as the QuickTime or DivX players use the settings of the MPEG-4 codec and the movie format and do not curtail the bitrate. If playback content is too complex to be mastered at full frame rate, the players drop frames. The DivX player incurs a significant jitter because it only notices the timing loss after the frame has been rendered, and CPU cycles that could have been used to render the next frame have just been wasted. However, both players can be performance-tuned by changing the codec settings while leaving the movie file untouched. The Windows Media Player reduces the content complexity within a few frames, so that jitter and frame loss occur only briefly (adaptive bitrate adjustment sometimes called *elastic* bitrate). As useful as these procedures are, they make it hard for the user to tune the trade-off between the available performance and the image.

The long startup time and the huge computing requirements require one to justify when to use MPEG-4. A small 6 MB MPEG-4 file included into PowerPoint did consume an initial startup delay of 2.5 s, allocate 28 MB of system memory, and cause 78% CPU load while playing. Due to the serial coding of the format, however, memory allocation will not exceed some 40 MB, even for long movies. Also startup times will not vary with file size. However, the decoding effort prior to anything being displayed can be cumbersome during a presentation, as can the high CPU load.

The use of MPEG-4 in microscopy should therefore be restricted to what it does best: playing long, highly compressed movies at guaranteed video rates. Like MPEG-2, MPEG-4 can play back an unlimited number of frames from the disk without interruption. Unlike MPEG-2 however, it eats many more CPU cycles and the number depends strongly on the movie content. The complexity of the MPEG-4 resource requirement is touched in the benchmarks listed below. Of more concern, one cannot be sure that an MPEG-4 file that runs smoothly on one system will perform well or even acceptably on a slightly less-powerful system. The only recommendation we can give here is to play the movie on your test system at maximum quality and run the CPU meter (in the TaskManager in Windows systems). Check that the CPU load does not exceed that of the presentation system. Be sure to leave plenty of margins when planning to play an MPEG-4 file this way. In contrast to MPEG-2 hardware, MPEG-4 accelerators are still scarce and — to make things worse — depend on the player software.

MPEG-4 exists in two completely different levels of operation: the more widespread H.263 level, referred to as MPEG-4 part 2, and the widely hailed H.264 (MPEG-4 part 10) (Richardson, 2003). H.263 achieves its very high compression rate by using a plethora of different compression techniques that all rely on certain assumptions about the movie content. For example, H.263 can encode 2D scene rotation, panning, dimming, and even object rotation without storing any image information whatsoever. Sadly, few of these abilities matter for micrographs.

On the other hand, H.264 deploys a new psycho-visual coding technique that is more effective than MPEG-2 but abandons the scene understanding of H.263. Therefore, it compresses less than H.263 but is also less prone for artifact creation. As a consequence, H.263 only works well in microscopy in sparse scenes and when the moving objects are not too small. When maximum compression is not essential there will be no serious use for H.263 in microscopy.

Because modern Apple systems support the H.264 decoders as standard,[7] one cannot use such a system to assure your movie will play well on an inferior machine.

For PCs, the situation is less encouraging but easier to control. Although later Radeon or X-series cards from ATI support the decoding of MPEG-4, they do so to a lesser degree than the Apple H.264 decoders but still boost performance. To get a better estimate, the hardware support for the MPEG-4 codec can be switched off and hence one can get a more reproducible performance estimate.

MPEG Display Formats

The most critical drawback of the MPEG formats is that they come in very few display raster sizes. MPEG-1 is available only in 352 × 288 resolution and hence limited to coarse VHS quality movies only. The upside is that it can be universally played and is not very computing power hungry (decoding effort about one third of main format MPEG-2, that is also well within modest computer limits). The so-called *main profile* of MPEG-2 is defined for the resolutions (called *levels*) 352 × 288 (low), 720 × 576 (main), 1440 ×

[7] H264 offers the same seamless playback and guaranteed frame rates as MPEG-2 does but at roughly half the bandwidth.

1152 (high 1440), and 1920 ×1152 (high) and for two frame rates 30 Hz (NTSC) or 25 Hz (PAL). There are three more MPEG-2 profiles but they offer no additional resolutions or frame rates. The hardware installed on most new graphics boards decodes the main profile but does not go beyond the 720 × 576 resolution. If your movie can be played back at one of these two frequencies or an integer fraction of it, then MPEG-2 offers low load for the computer, universal playback, and color calibration for many systems, and can be written to a DVD. In a presentation, it still may not be wise to play a 2 frame/s (fps) movie as an NTSC MPEG-2 because doing so may inflate a single frame to 15 frames. Also, as the jitter measurement in Table 48.1 shows, it can be cumbersome to encode a 27 fps movie to either PAL or NTSC. For the same reasons that most projectors offer a 30 Hz refresh rate, the NTSC format in a presentation will often play more fluently than the PAL format. If the movie is played full screen, then this limitation does not apply as the refresh will be switched to the 25 Hz needed for PAL.

The MPEG-4 format allows for arbitrary formats but playback software will not support deviations from the MPEG-2 levels. The codecs we tested would work on the main and the low MPEG-2 level for both MPEG-2 and MPEG-4. None would support the two high levels, and one MPEG-4 codec even crashed when using alternate resolutions.

The MPEG standard (Watkinson, 2000, pp. 9–16, 47) assumes that the analog image data is filtered in such a way that it cannot change more rapidly than the resolution that the digital movie contains.

In micrographs, this can be assured by not under-sampling the images. Only those digital micrographs that are properly sampled and not too noisy constitute suitable targets for either JPEG or movie compression algorithms. Unlike JPEG series, high-quality MPEG compression also requires proper temporal sampling to avoid compression artifacts. Consequently, MPEG compressors incorporate digital smoothing filters to handle noisy source images and, when an MPEG-compressed animation appears unnaturally sharp or hard edged with weak ripples and creases, this is a limitation of the standard and not of the player software.

Very High Resolutions

Except for the new H.264 standard, there is no widespread standard for playing movies at 1000 pixel square or higher resolution as is needed by microscopists. The other MPEG formats are practically limited to the MPEG-2 main level, the DVD's resolution. Although QuickTime can assemble single frames of any resolution in use today into movies (there is a filesize limit and a 2^{14} total pixels resolution limit) apart from the dated QuickTime compressor, it does not interface to any advanced movie standard to compress and store the result.

Microsoft's AVI format offers some of the same capabilities but with the even worse restriction that they provide no compressor at all and merely accommodate playback. Movie authoring software such as Adobe Premiere, Final Cut Pro, or Jasc AnimationShop can write megapixel movies, and so do the tools of most microscopy software. When it comes to compression ratios, QuickTime performs better than the antique Sorensen or Cinepak coders used to support the high resolutions for AVI. Apple Final-cut Pro can now create HDTV MPEG-4 (H.264) and H.262 movies.

Movie Compression and Entropy

To obtain some generally valid results and establish a concept of movie performance and quality, it is necessary to focus on the complexity of the movie. The more information a movie conveys (not necessarily the same as **useful** information!) the harder it is to compress, to decompress, and to play back in real time. Complex movies will both suffer from lower frame rates and more compression artifacts. In addition, they will not compress as well. Therefore, the compression factor can be used as a scale for the movie complexity. The compression depends on the entropy of the movie information — the more random the images and animation are, the less it can be compressed without apparent degradation. To get "clean" measurements, we have used an image generator that yields image sequences in which the randomness is controlled. As a result, the image or movie content will be perfectly random to test the movie format but the local changes in the image and the fluctuations in time will be more rapid with higher entropy. In information theory, one can either use this "channel entropy" when talking about compression or when assessing it by taking the autocorrelation of the image sequence. This measure can be scaled to 1 for a completely constant image sequence and 0 for one where neighboring pixels have no relationship to each other in space and time. To benchmark the players and the formats, we devise three movies, one with autocorrelation 0, that is, completely random signal (Format A). One with the highest useful complexity, that is, a movie that is Nyquist sampled in space and time but is as random as possible within these restraints (Format B). And finally, a sparse movie as they are popular in fluorescence labeling, with a real two-channel sequence forced to proper sampling in space and time (Format D). Very often, temporal and spatial sampling is not performed properly. Because, despite all efforts to preach the ultimate importance of proper sampling, under-sampled material still enjoys popularity, we also include an under-sampled version of Format D as Format C.

For the compressor, A is harder to handle than B, B harder than C, and C harder than D. Instead of some meaningless autocorrelation factor, we give the compression ratio (q) as a percentage, that is, the size of the compressed movie file divided by the size of the uncompressed image series. Unfortunately, decompression mostly behaves inversely and a well-compressed movie (low q) will take more computer cycles to play than a more complex one that is compressed less. The exceptions are the MPEG-2 and H.264 formats with a variable bitrate, as here compression and playback effort is symmetric. Because they all guarantee 25 to 30 frames per second, we do not include playback speed in the benchmarks.

Performance Benchmark

Speed benchmarks give you a very coarse idea of what is doable in movie animation. We intentionally do not use the latest Nvidia GeForce6 series or the ATI X600 and X800 engines that are common in G5 Apple machines because we need to get the presentation running on an average system available today. Hence, we test a notebook with a 2-year-old GeForce4MX mobile engine and a desktop with a GeForce 5600. Both cards do not support hardware movie features for non-MPEG playback and hence we should obtain movie performance that is somewhat related to system power.

Compression Ratios for TV-Sized MPEG Movies

VBR is variable bitrate, that is, a compression that adapts to the movie content. CBR stands for constant bitrate, meaning that the compression ratio is held constant, independent of the content.

TABLE 48.2. Compression of TV-Sized Movies (720 × 576)

Format/Type	A	B	C	D
MPEG-2 CBR	2.3%	2.3%	2.3%	2.3%
MPEG-4 CBR	0.8%	0.8%	0.8%	0.8%
MPEG-2 VBR	2.5%	2.5%	2.1%	2.1%
MPEG-4 VBR	2.0%	2.0%	1.7%	1.7%

TABLE 48.4. Playback Frames per Second for PAL TV-Sized Movies (30-s Window)

Format/Type	720 × 576 A	720 × 576 B	720 × 576 C	720 × 576 D
QuickTime NC	27 fps	27 fps	27 fps	27 fps
QuickTime JPG	22 fps	22 fps	24 fps	24 fps
QuickTime QT	32 fps	34 fps	34 fps	34 fps
AVI Cinepac	60 fps	60 fps	60 fps	60 fps

Please note that the two modes cannot be compared to each other as they serve completely different quality needs. CBR is used when there is only a limited bandwidth that can be transferred. VBR is used when one needs at least a certain quality level. The settings used here are default settings for 2-h DVDs (M2 settings), resulting in a 5000 to 6000 kbit/s bitrate. The CBR ratios are trivial as the movie size is given via the bitrate. A high-quality DVD movie would use around 6000 kbit/s. The limit of the MPEG-2 standard (even for VBR movies) is 15,000 kbit/s, including all soundtracks. The MPEG-2 compressors one can buy generally cap at 9200 kbit/s. Commercial DVD movies will not go below 4000 kbit/s.

The raw movie material assumed in developing Table 48.2 would consume a datarate of 30 Hz × 720 width × 576 height × 8 bits per sample, that is, 298,500 kbits/s.

MPEG-4 goes far below this mark. Movies are often encoded at 600 to 700 kbit/s and 1500 kbit/s is considered high quality.

These numbers (Tables 48.3–48.6) list file size and frames per second of different formats. Determining quality involves examining the artifacts introduced and for most rich micrographs, the uncompressed and motion JPG formats may very well be the methods of choice, despite their lack of speed and high memory requirements. Cinepac may be a good and universal choice for a Web page but hardly ever lives up to the needs of displaying micrographs.

Important note: the numbers for QuickTime are QT 6.5 benchmarks, QT 5 actually has higher frame rates for the uncompressed case (17.5 fps) and identical rates for motion JPEG. Because QT 6 (and higher) is more efficient while compressing and supports MPEG-4, we use the latest version.

As in the case of displaying static images, one must realize that the presentation system may have a different screen resolution from the system you tuned the movie on. In this case, the content must either be zoomed or — much more likely — downsized to match the projection system. Respect the restrictions on down-sampling given early in this chapter and also keep in mind that down-sampling will cost CPU power, though less than 5% on a 2 GHz system. If you must run your animation on an unknown computer, make sure that you have plenty of leeway.

TABLE 48.3. Compression Ratios for PAL TV-Sized Movies (720 × 576)

Format/Type	*q* of A	*q* of B	*q* of C	*q* of D
QuickTime NC	100%	100%	100%	100%
QuickTime JPG	27%	18%	3.5%	3.5%
QuickTime QT	17%	18%	3.5%	3.5%
AVI Cinepac	7%	7%	5%	4.5%

Storing Your Presentation for Remote Use

Besides texts and drawings, Keynote and PowerPoint presentations can include image and animation data as well as the complex scripts needed to present them. Images and animations can either be stored within the presentation or they can be linked to the latter by reference. In the latter case, PowerPoint or Keynote will retrieve the original image or movie by a path and file name (the link), and then display it by rules stored within the presentation. The display rules at a minimum consist of dimensions and positions but can also contain timing and movement, triggers that react to certain other events or scripts.

Due to the native viewer concept, PowerPoint hosts images in their original format and hence does not compress or reformat the image data for storage. Any resizing and resampling is done only when the image or movie is displayed. This means that any image included in a PowerPoint presentation will increase the PPT file by the size of the original image (plus a base overhead of about 4 kB for the displaying rules) and will have to be decompressed and resized each time it is displayed in a slide. The data of each image is then cached by PowerPoint so that, the display is faster, the second time it is shown, even though the image must still be decompressed each time. As a result, it is better to only use images that have no more resolution than is actually needed.[8]

There are several reasons for only storing references within a presentation. The premier one is that only a single copy of an image or a movie has to be stored no matter how many different presentations may use it. This also assures update consistency, as changes to a file will be available in all the presentations that include it.

The main drawback is that keeping the references consistent when transporting a presentation to a different computer is not trivial because the path referenced in the document must also function properly on the presentation computer.

There is no perfect solution to this situation and all approaches are plagued with obvious shortcomings:

1. PowerPoint offers a ***pack-and-go*** mode. The file made using this command includes the images and animations (obtained from their references). This is the fastest (and fairly safe) way to complete the task but it leaves one with a single-use presentation. Should you perform changes and add material to the presentation, it will be cumbersome to revert it back into a presentation with links again. The inclusions have to be deleted manually and replaced by new, valid links. So if there is the chance that you will want to update

[8] An exception might be if one is using Keynote or PowerPoint not primarily for a presentation but instead as a way of storing a number of figures for a printed publication. This can be a convenient means of accumulating and annotating the figures and in this case, you can store the images at the resolution appropriate for their final use in the article.

TABLE 48.5. Compression Ratios for Large Movies (1024 Square)

Player/Format	*q* of A	*q* of B	*q* of C	*q* of D
QuickTime NC	100%	100%	100%	100%
QuickTime JPG	24.2%	15.1%	2.75%	2.75%
QuickTime QT	16.6 %	16.7%	2.9%	2.9%
AVI Cincepac	5.2%	5.2%	4.1%	4.0%

your presentation after packing, this approach will not work well.

Also, failures of this procedure have been reported. It works flawlessly for images and PowerPoint animation software but large movies that may be processed by codecs not originally shipped with Windows, sometimes do not play. This is particularly common with movies formats that contain references in themselves such as do MPEG-4 movie streams. Even under OSX, non-standard codecs can lead to incomplete wrapping of the presentation.

The danger here lies in the fact that this situation cannot be avoided by packing then testing the presentation, as the references on the machine the packing is performed on are still valid and failure will only occur once the presentation is loaded onto the target computer. A safe way to force a valid test is to unlink the references — if all your included data is distributed in a path (that would be a directory name in Windows or a folder in OSX) say "all," rename "all" as "former_all." Then running the packed presentation will actually invalidate improper links and the error will show when the presentation is run.

2. ***Moving the entire filesystem*** (or file tree) is a very flexible option for people who try to avoid version conflicts between presentations. If you keep all the data you work with and intend to present in a certain path (a drive and directory tree under Windows or a mounted folder under OSX), then you will be able to move this file system between different computers and keep all references in your presentations valid. For Mac users, this goal can be achieved almost trivially by having a portable disk with a name that does not occur on any of the target computers. So when the disk with the cryptic name is plugged to an alien Mac, it will appear as a folder accessible from the desktop. All presentations can then be moved without any changes.

PCs have an unpleasant restriction that the drive name assigned to such a disk must be free on the target computer. So it is not wise to use any disk drive letter either between A and E, as they tend to be occupied by disks, optical drives, and memory sticks, or using the last characters V through Z as they may also be used for images of network drives. Other than that, the more remote letters work reliably and using a drive letter far from these zones of confusion usually works well.

Newer operating systems, such as Windows 2000 and XP, allow the use of "soft links" similar to the folder concept of the Apple operating system. You can link your portable drive with a soft link from the desktop and when transporting presentations simply re-create that soft link on the target machine. This approach will not work reliably with dated PowerPoint versions as soft links are often resolved to their absolute paths. Given that you use a drive X: with all your presentations, hence using the basic path X:\all_my_presentations, you could soft link it from the desktop, say as "all_my_presentations." PowerPoint 97 may still store the references with the drive name though and hence limit portability. However, as long as this link name is not in use on the target system, this provides a very reliable transportation mechanism for newer PowerPoint versions.

3. Approach 2 won't work if either you cannot use your own disk or want to hand out a presentation. Sharing a presentation is made much easier if the sharing is planned in from the start. If all images and animations for a presentations are made available in a single directory — or folder — from the beginning, then simply copying that folder will make the presentation mobile. This approach is referred to as a ***flat file structure,*** in contrast to the directory tree which is hierarchical. For older PowerPoint versions (especially the still-popular 95 and 97 releases), this can lead to a soft error the first time the presentation is run and PowerPoint will feel obliged to ask each reference for an updated location. The newer versions, however, will deal with the flat file structure smoothly.

Note, however, that with this approach, the person who receives the presentation has copies of the images and movies available in native format. Whether or not you want to give them away, PowerPoint itself does not prevent your data from being extracted. PowerPoint files are documented and even a pack-and-go presentation can be disassembled into its components again.

To obtain more sophisticated protection of your images and animations, make sure to incorporate a copyright watermark into the scans and the movies. Adobe PhotoShop and Premiere, as well as some plug-ins for QuickTime, will perform this task. Watermarks are very difficult to remove with current tools. Another strong protection measure is to lower the resolution of the image material. Scans reduced to half their size, say 512 square will look fine on a projector but will look inferior when printed. Animations in MPEG-1 format can even look great despite the fact that they are of far lower resolution (352×288 pixels at native resolution) than the time series from which they were obtained (usually of megapixel resolution). Both measures, watermarks and lowered resolution, provide the best available protection of your data at this stage. More advanced protection algorithms, such as steganography, which encodes additional information into file formats, is sometimes removed when "saved again" on some tools. Field marks or any other watermarks that cover the entire image are very safe but do impair the image quality. They imprint a noise pattern on the image that is unique to a password you choose and the pattern is chosen in a way that standard filters, such as smoothing or noise reduction, will not remove. In contrast to normal watermarks, however, it is not visible to the eye and hence has no meaning for daily copyright issues. It also does not survive reliably in print and so will not stop others from using it.

In Apple Keynote 2, the Save As command yields a window that asks if you want to copy theme images, audio, or movies into

TABLE 48.6. Playback Frames per Second for Large Movies (30-s Window)

Player/Format	1024 × 1024 A	1024 × 1024 B	1024 × 1024 D	1024 × 1024 D
QuickTime NC	11.5 fps	Same	Same	Same
QuickTime JPG	7.0 fps	8.0 fps	8.0 fps	8.0 fps
QuickTime QT	16.0 fps	15.5 fps	17 fps	17 fps
AVI Cinepac	26.5 fps	27.0 fps	25 fps	25 fps

the document and this seems likely to produce a file similar to the PowerPoint pack-and-go file.

Taking Your Presentation on the Road: Digital Rights Management and Overlaying

When playing animation formats, some advanced features of the native player software can cause substantial inconsistencies. All players are optimized to provide high performance and will exploit hardware features if the drawing libraries (DirectX and QuickDraw) allow this. One of the very popular features shared by the two main players in the field — literally, for once — is dubbed overlaying. For this, the content of the movie frames is not drawn directly into the video memory but into a secondary buffer named the overlay. When the graphics engine refreshes the display, it reads the contents of the overlay, if one is present, rather than the contents of the main image buffer. Overlays must be of rectangular shape and the hardware places strict limits on how many overlays there can be in a display frame. The limit can actually be as low as 1 or 0. The advantage of doing this is that the display does not have to be redrawn when the movie plays. This permits a much more naturally integrated movie window that will respond to scaling and moving much like any other application, and that hides its thirst for computing power.

However, on some hardware, overlaying becomes a problem when a projector is added in parallel to the primary display (i.e., on a laptop, the number of overlays for desktop machines is often more than 1 and hence the problem may not arise — repeat "may"). Under these conditions, sometimes the overlay only works for the primary display and the movie output on the projector is blank (actually, black).

Overlaying can be turned off in MediaPlayer: choose Tools → Options, switch to the Performance tab and choose Advanced, and there, under Video Acceleration, you can disable Use Overlays. There is a second overlay switch for DVD playback below the first one that you might have to deactivate if you play your movie from a DVD.

And in QuickTime: go to Edit → Preferences → QuickTime Preferences, toggle to Video Settings, and deactivate Enable DirectDraw.

As this will disable all acceleration via the graphics library and hence impair performance, it is not the generic setting to be used all the time. It is just a possible solution to the blank-movie window problem described above.

A much more general problem that seems to be dramatically increasing over time is the Digital Rights Management mechanism built into the standard players. This is a protection mechanism intended to uphold copyrights and inhibit duplication of copyrighted material and is capable of preventing you from grabbing screen shots and sequences (clips) from DVDs or other restricted movies.

The problem is that, as there is no open publication of what DRM does and how it affects your system, the normal user can have problems when the player interprets your movie as being copyrighted even when it isn't. This type of error can happen when using advanced formats[9] in QuickTime and MPEG2 and the results can vary from not being able to play the movie in PowerPoint to just not being able to display it on a projector.

A lesser nuisance is that movie startup may be delayed while the player tries to check to see if it is authorized to play the format involved.

The safest way to prevent problems of this kind is to keep your movies in the original format when playing them. Even playing a DVD as part of a presentation may lock the player to a fixed frame rate and format. If the movie files are present in the original, simple format (MPEG files or AVIs that you wrote yourself), it is very unlikely that you will encounter these problems.

Finally, resist the temptation to update the version of your native players just before a presentation. Continuously escalating DRM requirements may make unplayable a movie that had been playable on an earlier version. Although the chances that this will happen are small, they are fatal for a presentation. In general, it is inherently risky to update your player as some of your movie formats may no longer be supported or the new player may insist that your formats must comply with certain rules and will not play them unless they do. This applies particularly to QuickTime files and MPEG-2 or MPEG-4 formats.

At present, most player software still behaves benignly. They will play all formats to the projector if the overlaying is switched off, and when they cannot retrieve a license, they search for a movie file, and then play this format. One notable exception involves movies on DVDs, and these come with severe restrictions and may not even be able to be played at the frame rate of your projector. However, the topic of DRM continues to evolve quickly and, at present, the only way to retain control over your animations is to keep them in the original formats that you can edit (i.e., the production format). In other words, you can use the QuickTime, MPEG-4, and MPEG-2 formats, but you must keep them original (i.e., after writing them, do not package them or write them onto a DVD or VCD), unless you are sure that you have not included or activated an any copyrights when producing the movie.

If you do have to resort to DVDs, make sure that you do not create a *Region Code* that will prevent the DVD from playing on another continent than the one on which the DVD was created. This problem can be a particular nuisance when assembling presentations for use at international conferences. The DVD feature to avoid at all costs is *User Prohibitions*, that is, locking the way the DVD has to be played and accessed. These constraints will prevent the player from playing even an excerpt from the movie and they will also lock down the frame rate of the playback. Unless you can make sure that these features are switched off when creating your DVD, you run the risk that others may not be able to play your presentation and you, yourself, may not be able to play it on any other system or, at least not in the sequence you intend to. To be prepared, it is always wise to carry a copy of your production files with you.

Good Luck!

HELPFUL URLS

The TIFF Standard Archives

http://www.digitalpreservation.gov/formats/fdd/fdd000022.shtml

Nvidia Graphics Cards Technical Details

http://developer.nvidia.com/page/home

[9] If you use an animation editing tool such as Premiere, you can copyright the output material which gives you a different file format from the original data. Editing lingo is production format for the original file and consumer format for the distributed, that is, the copyrighted file. These latter files are susceptible of being handled more strictly by DRM mechanisms.

ATI Graphics Card Technical Details

http://www.ati.com/developer/index.html

Video and Image Compression Lingo Site

http://streaming.wisconsin.edu/terms.html

Wikipedia Encyclopedia

http://en.wikipedia.org

The OpenGL Committee and Graphics Language Reference

http://www.opengl.org

General

http://developer.apple.com/
http://www.vitecmm.com/
http://www.jpeg.org/

ACKNOWLEDGMENTS

Thanks are owed to Valentin Guggiana, ETH Zurich, for his help with setting up benchmarks on systems I do not comprehend; to Dr. Chris Pudney of Focal Technologies for his recommendation and advice on software tools and standards; to Dr. Urs Moser and Felix Gattiker of Sulzer Innotec for providing image material, image-processing software, and numerous hardware platforms to test our assumptions; to Dr. Guy Cox, University of Sydney, for advice on the chapter and providing material from his own chapter; and, finally, to Dr. James Pawley, University of Wisconsin, as the concept and the structure of the chapter and most of its final outcome were conceived by him.

REFERENCES

Castleman, K.R., 1995, *Digital Image Processing*, Prentice Hall, Englewood Cliffs, New Jersey, pp. 247–248.

Bracewell, R., 1995, *Two Dimensional Imaging*, Prentice Hall, Englewood Cliffs, New Jersey.

Ebrahimi, T., and Pereira, F., 2002, *The MPEG-4 Book*, Prentice Hall, Englewood Cliffs, New Jersey.

Nikolaidis, N., and Pitas, I., 2000, *3-D Image Processing Algorithms*, John Wiley and Sons, Inc., New York, pp. 20–24.

Richardson, I.E.G., 2003, *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*, John Wiley and Sons, Hoboken, New Jersey.

Watkinson, J., 2001, *MPEG Handbook*, Focal Press; Linacre House, Oxford, UK.

Walsh, A.E., and Bourges-Sévenier, M., 2002, *MPEG-4 Jump-Start*, Prentice Hall, Englewood Cliffs, New Jersey.

Woo, M., Neider, J., and Davis, T., 2000, Texture mapping, in *OpenGL(R) Programming Guide*, Addison Wesley, pp. 351–466.