Spatial Vision, Vol. 10, No. 4, pp. 437–442 (1997)

The *VideoToolbox* software for visual psychophysics: transforming numbers into movies*

DENIS G. PELLI**

Department of Psychology, New York University, 6 Washington Place, New York, NY 10003, USA

Received 1 February 1996; revised 8 September 1996; accepted 8 September 1996

Abstract—The *VideoToolbox* is a free collection of two hundred C subroutines for Macintosh computers that calibrates and controls the computer-display interface to create accurately specified visual stimuli. High-level platform-independent languages like MATLAB are best for creating the numbers that describe the desired images. Low-level, computer-specific *VideoToolbox* routines control the hardware that transforms those numbers into a movie. Transcending the particular computer and language, we discuss the nature of the computer-display interface, and how to calibrate and control it.

1. INTRODUCTION

Using a high-level platform-independent language like MATLAB, it's easy to produce a matrix of numbers specifying the desired luminances of all the pixels in the displayed image. Many of today's off-the-shelf personal computers can copy those numbers from memory to video memory quickly enough to show a new image on every frame of a CRT monitor. However, high-level languages generally provide only rudimentary control of the vital transformations from number to color, and of the rate at which successive images are displayed. That is where the *VideoToolbox* comes in, providing tools to measure and control the pixel transformation, and timing synchronization of the computer-display interface. The *VideoToolbox* is a free collection of two hundred C routines for Macintosh computers that make it possible to display accurately specified visual stimuli. Thanks to the efforts of Brainard (1997) and Solomon and Watson (1997), it is now possible to use the *VideoToolbox* from within MATLAB and Mathematica. Here we discuss the heart of the computer-display interface problem, which transcends the particular programming language and computer operating system.

Once the matrix of numbers has been loaded into video memory, the subsequent transformation from number to luminance (or color) is complicated, but usefully

*http://rajsky.psych.nyu.edu/VideoToolbox

**denis@psych.nyu.edu

simplified to three steps. First, at video rates (e.g. 100 million pixels per second), each number passes through a lookup table, typically one 8-bit number in and three 8- to 10-bit numbers out, each driving an 8- to 10-bit digital-to-analog converter. Second, the three analog video signals drive the three guns of a color CRT. Luminance is proportional to beam current, which is an accelerating function of drive voltage: the monitor's gamma function. Third, the luminous image is blurred by the point-spread function of the beam. The luminance and point spread vary with screen position, but for many vision experiments it is enough to characterize the center of the screen. The *VideoToolbox* includes programs to measure the gamma and point-spread functions, and subroutines to compute an optimal lookup table to do gamma correction (see Pelli and Zhang, 1991) and to load that table into the video hardware. The measured point spread, or, more precisely, the spatial-frequency modulation transfer function (MTF) of the monitor, should be taken into account in the user's image synthesis software.

2. SYNCHRONIZING

It is surprisingly difficult to synchronize a computer program and video card. Like computers, most video cards have autonomous free-running clocks, and in the video industry are called 'masters', expecting the video monitor to be a slave, following its timing. In experiments for which stimulus timing matters, the user program will normally have to synchronize itself to the video card, by waiting for each frame to end.

In Macintosh computers and, increasingly, in most other computers well, user programs cannot access the video card hardware directly, because it is undocumented. Instead, user programs are allowed to send requests to the video driver associated with the video card. The video driver is a special program written by the manufacturer of the video card that provides a standard interface, accepting a few standard calls specified by the operating system. The only video-driver call that we use frequently is the one that loads the video card's lookup table.

There are four ways to synchronize a program to a video card.

1. Use the vertical-blanking interrupt, which occurs once per frame, at the beginning of the vertical blanking. Typically, your interrupt service routine should just increment a frame-counter variable; have your main program use a wait loop to synchronize itself to the frame count. A disadvantage of this approach is that it fails if you raise the processor priority to block all interrupts, which is a convenient way of temporarily suspending all irrelevant activity (e.g. network traffic) to make all the computer's time available for your experiment.[1]
2. Most video drivers, when asked to load the lookup table, wait until the vertical blanking interval before beginning. (They wait in order to avoid creating visible hash on the screen while the lookup table is changing.) This has the side effect of synchronizing your program to the display, since the driver does not return control until shortly after the vertical-blanking interval begins.[2]
3. Most video cards provide a read-only vertical-blanking bit (telling whether the video signal is in the blanked period between frames), but unfortunately

few manufacturers will tell you its address. Knowledgeable engineers have confirmed to me that indeed most video drivers monitor this bit to wait for the blanking interval. An advantage of this approach is that your program can determine not only when blanking begins, as in methods 1 and 2, but also when it ends.[3]

4. Use an analog or digital input to sample the video signal, and watch for the leading edge of the vertical retrace pulse (Dave Post, personal communication).

3. WHAT ELSE?

The *TimeVideo* application checks the timing of all video devices in anticipation of their use in critical real-time applications, e.g. movies or lookup-table animation. Low-level routines control video timing and lookup tables, display real-time movies, and implement the luminance-control algorithms suggested by Pelli and Zhang (1991). In particular, *CopyWindows* (or *CopyBitsQuickly*) faithfully copies between on-screen and off-screen windows (or bit/pixmaps), *WindowToEPS* saves an image to disk as encapsulated PostScript, for later printing or incorporation into a document, and *SetEntriesQuickly* and *GDSetEntries* load the screen's color lookup table, all without any of *QuickDraw's* color translations. *NoisePdfFill.c* quickly generates visual noise images whose pixels are samples from a specified probability density function. High-level routines help analyze psychophysical experiments (e.g. maximum-likelihood fitting and graphing of psychometric data). *Assign.c* is a runtime C interpreter for C assignment statements, which is useful for controlling experiments and sharing calibration data. This collection has been continually updated since 1991. Two hundred colleagues subscribe to the email notification of updates. Many of the routines are Mac-specific, but some very useful routines, e.g. the luminance-control, statistics, maximum-likelihood fitting algorithms, and the runtime interpreter are written in standard C and will work on any computer.

The *VideoToolbox* web site, in addition to the software, displays current information about relevant hardware and software, e.g. 'Monitors' and 'Video cards', and advice about how to program experiments. 'Video bugs' reports all known bugs uncovered by *TimeVideo* testing of 56 video cards and drivers.

4. MATLAB, MATHEMATICA, AND RSVP

Brainard's (1997) *Psychophysics Toolbox* for MATLAB makes it easy to use the *VideoToolbox* from within MATLAB. Solomon and Watson (1996) have created a Mathematica package *Cinematica* containing functions, based on the *VideoToolbox*, that produce calibrated grayscale movies on a Macintosh from within the Mathematica programming environment. (Also see Watson and Solomon, 1997.) The Tarr and Williams (1996) *RSVP* program, also built using elements of the *VideoToolbox*, is an extensible software package for running psychology experiments.

5. VIDEO ATTENUATOR

The typical 8-bit resolution of video digital-to-analog converters (DACs) is inadequate to render threshold-contrast stimuli, typically yielding on-screen contrasts that are only accurate to about 7 bits, because of the monitor's factor-of-two contrast gain. Pelli and Zhang (1991) describe a simple electronic circuit, a video attenuator, that provides accurate contrast control, achieving 12-bit on-screen accuracy from any 8-bit color DAC. A passive six-resistor network combines the three color RGB signals from a color video card to produce a single monochrome signal of higher grayscale resolution. The *VideoToolbox* includes software to do gamma correction and takes full advantage of a video attenuator.

6. WHAT'S THE *VideoToolbox* GOOD FOR?

Users of the *VideoToolbox* say (see *Acknowledgments*) 'it turns the Mac's video into a scientific instrument', 'providing a simple interface to the guts of the Mac,' to 'calibrate resolution and timing' and 'produce all sorts of visual displays', especially 'calibrated grayscale movies'. Its 'great 'snip and use' library' 'is reliable and incredibly versatile'; 'each routine can be used à la carte.' With its 'priceless advice on setting up' and 'invaluable hardware notes,' 'it's a great way to get up and running on a Mac.' In sum, it's 'good for getting the Mac to dance to the tune of psychophysical stimuli without having to learn all the steps yourself'.

7. AVAILABILITY

VideoToolbox is available at the web site indicated on p. 437, or from the Info-Mac archive; search for 'video-toolbox'
http://hyperarchive.lcs.mit.edu/HyperArchive.html

Psychophysics Toolbox for MATLAB
http://www.psych.ucsb.edu/~brainard/software/

Cinematica for Mathematica
http://vision.arc.nasa.gov/mathematica/Cinematica/Cinematica.html

RSVP
http://www.cog.brown.edu/brochure/people/mjt/projects/rsvp.html

Acknowledgments

We thank Fernando Urbina of Apple Computer for customizing the 7500/7600/8500 video driver for vision researchers who need high frame rates, and we thank our many friends and colleagues who have contributed to the *VideoToolbox*:

Mike Alexander on why you can't replace the boot monitor's driver; Sigurdur Asgeirsson on the need to zero *tmReserved* in *Timer.c*; Michael Bach on interrupts and

analog data acquisition; Paul Beckmann on monochrome monitors; Kevin Bell for *PatchExitToShell* in *Timer.c*; David Brainard on synchronizing and for *AfterDark.c*, parts of *Assign.c*, part of *GDOpenWindow.c*, *GetTimeDateString.c*, and *PeekTimer* in *Timer.c*; Tom Busey on producing stereo; E. J. Chichilniski for *SetFileInfo.c*, Raynald Comtois on spurious interrupts and for *SetEntriesQuickly.c*; Frans Cornelissen for the *VideoToolbox* folder icon; Steve Coy for *AtExitToShell.c*; Michael Eckert on 7100 NuBus bug; Rhea Eskew on *Psychophysics Display Interface*; Bart Farell on color cycling and for several routines in *SetOnePixel.c* and *SetPixelsQuickly.c*; Bill Haake for *SetEntriesQuickly.c*; Lance Hahn on analog data acquisition; C. K. Haun for *KillEveryoneButMe.c*; Bill Hofmann for *AtExitToShell.c*; Mike Kahl for *CopyQuickDrawGlobals.c* and *kbhit.c*; Sergei Kurkin on *GDSetPageDrawn* and *GDSetPageShown*; Joseph Laffey for *GetVersionString.c*; Steve Lemke on *ShieldCursor*; Peter Lennie for *SetEntriesQuickly.c*; J. N. Little and Jim M. Boyles for *ReadMATLABFile.c*; Jamie R. McCarthy for *IsCmdPeriod.c*; Harry Orbach on bright fast monochrome monitors; Izumi Ozhawa for *CVNetConvert.c*; Flip Phillips on Mac OS 7.6 drivers; Dave Radcliffe for *FlushCacheRange.c*; Evan Relkin for *kbhit.c*; Tom Robson on video monitors; Mike Schechter for *PixMapToPICT.c*; Dan Sears for *MoveMouse.c*; Eric Simenel on need to zero *tmReserved* in *Timer.c*; Paul Sowden on video bandwidth; Stefan Treue for suggestions and corrections to *Timer.c*; John Troy on Sony monitors; Preeti Verghese for *GetVoltage.c*; Karsten S. Weber on synchronizing two video cards; and Wei Xie on *SetEntriesQuickly.c*.

The *VideoToolbox* 'Video bugs' lists all known bugs in Macintosh video drivers, based on *TimeVideo* reports from: Ken Alexander, Mike Alexander, David Brainard, Tom Busey, Kyle Cave, Michael Eckert, Kaan Erdener, Rhea Eskew, Bart Farell, Patrick Flanagan, Mike Garver, Tony Hayes, Gregory Jackson, Richard Kunert, Sergei Kurkin, Jan Linder, Jan-Eric Litton, Brian McElree, David Rose, Ariel Salomon, Dan Sears, Steve Shevell, Paul Sowden, Scott Stevenson, Fernando Urbina, Jack Van Olst, Marty Wachter, Beau Watson, Wei Xie, and Xuemei Zhang.

'What's the *VideoToolbox* good for?' above, is based on responses from: Ulf Ahlström, Michael Bach, Ben Backus, Curtis L. Baker, Thomas A. Busey, Russell Clarke, Larry Cormack, Andrew Derrington, Bob Dougherty, Richard Eagle, Bart Farell, Debbie Giaschi, Mike Harris, Lewis O. Harvey, Jr., Jim Huffman, Keith Karn, Lenny Kontsevich, Martin Lankheet, Peter Lennie, Ju Li, Don MacLeod, Pascal Mamassian, Iain Merrick, Sacha Nelson, Harry Orbach, David Rose, David Schloerb, Ken Scott-Brown, Michael Shadlen, Ben Singer, Joshua A. Solomon, Michel Treisman, Stefan Treue, and Preeti Verghese.

NOTES

1. Some video cards spuriously emit more than one interrupt per frame, apparently because they assert the interrupt line for too long. On Macintoshes you can check this by running the *VideoToolbox TimeVideo* diagnostic program.

2. Loading the lookup table is fast (transferring less than a kilobyte), so that several lookup-table-load requests in quick succession might occur during a single vertical blanking interval. If you want each request to wait for the end of a new frame it might be necessary to pause for about a millisecond between

requests. The *VideoToolbox TimeVideo* program compares the rate of back-to-back lookup-table-load requests with the frame rate.

3. I determined one video card's vertical-blanking bit's address by disassembling its video driver. A less tedious approach might be to write a program that could find the vertical-blanking bit automatically by looking for any bit that changes at the same frequency as the vertical-blanking interrupt.

REFERENCES

Brainard, D. (1997). The *Psychophysics Toolbox. Spatial Vision* **10**, 433–436.

Pelli, D. G. and Zhang, L. (1991). Accurate control of contrast on microcomputer displays. *Vision Res.* **31**, 1337–1350.

Solomon, J. A. and Watson, A. B. (1996). Cinematica: A system for calibrated, Macintosh-driven displays from within Mathematica. *Behav. Res. Methods Instrum. Comput.* **28**(4), 607–610.

Tarr, M. J. and Williams, P. (1996). RSVP: A software package for experimental psychology using the Apple Macintosh OS. Department of Cognitive and Linguistic Sciences, Brown University, Providence, RI.

Tyler, C. W. (1997). Colour bit-stealing to enhance the luminance resolution of digital displays on a single pixel basis. *Spatial Vision* **10**, 369–377.

Watson, A. B. and Solomon, J. A. (1997). Psychophysica: Mathematica notebooks for psychophysical experiments. *Spatial Vision* **10**, 447–466.