Behavior Research Methods, Instruments, & Computers
2000, 32 (3), 458-463

A low-cost, accurate method of producing large quantities of digitally filtered images

KENNETH C. SCOTT-BROWN and TIMOTHY R. JORDAN
University of Nottingham, Nottingham, England

We demonstrate how to produce complex image transformations of bitmap files for vision experiments using the Cogimatic Vision Starter Kit (VSK) library of mathematical routines along with Visual Basic, C++, or the Delphi Pascal compiler. Implementing this system on an IBM-compatible PC running Windows 95, 98, or NT4 enables researchers to quickly and economically manipulate images for vision research. The VSK includes a simple stand-alone image-processing application. In addition, VSK has the ability to automate image transformations and to fully integrate image processing into new experimental software on the PC platform.

The importance of spatial filtering is both long established (see, e.g., Braddick, Campbell, & Atkinson, 1978; Campbell & Robson, 1968; and Graham, 1989) and of contemporary importance to cognition and vision research (e.g., Alexander, Xie, & Derlacki, 1994; Fiorentini, Maffei, & Sandini, 1983; May, Brown, Scott, & Donlon, 1990; Olds & Engel, 1998; Parker, Lishman, & Hughes, 1992; Schyns & Oliva, 1997; Solomon & Pelli, 1994). In the present paper, we show how current computer technology can be used to produce filtered images quickly and cheaply.

Early studies (e.g., Legge, Pelli, Rubin, & Schleske, 1985) used ground glass diffusers at variable viewing distances to provide a low spatial frequency blurred image with different spatial frequency cutoffs. Hogben, Pratt, Dedman, and Clark (1996) used frosted acetate in a similar fashion. Subsequent research on the statistical properties of images (e.g., Brady, Bex, & Fredericksen, 1997; Parish & Sperling, 1991; Solomon & Pelli, 1994; Watt, 1991) have placed intensive demands on filtering techniques, requiring computational solutions that are both quantifiable and accurate. Programmable digital systems give the user precise, flexible, and replicable control over the properties of filters, and avoid the variability inherent in optical alternatives such as ground glass. However, the computational demands of producing complex filtered images require both mathematical sophistication and powerful computers such as Unix workstations to run sophisticated image processing packages (e.g., HIPS: Landy, Cohen, & Sperling, 1984a, 1984b).

Commercial desktop publishing packages with extensive possibilities for general artwork generation such as Adobe Photoshop are now available for PCs (and for Macs). However, their lack of programmability in a laboratory setting is disadvantageous, because user-generated stimuli modified by mouse-clicked menu operations are laborious to construct and may contain repetition-induced errors which are difficult to detect at both execution and verification stages.[1] These packages also are limited in their capacity to produce precisely defined parameter changes across successive images, because the available filters suit artistic needs rather than the technical requirements of vision researchers who may require fast Fourier transforms and relatively obscure filter shapes. Such packages are also costly.

Viable solutions for vision research are available on the Macintosh platform; for example, the public domain NIH Image program includes many of the filters used in computer vision applications plus a capacity for automatic processing via a macro language. However, one drawback of these macros is the lack of an integrated debugger, which restricts sophisticated use by making code development potentially more error prone and slower. The overall functionality of the macro language's commands is less powerful than a modern C++ or Pascal integrated development environment (IDE), both of which have a more extensive range of programming statements. Moreover, since Image is a stand-alone application, users are unable to actively link the output to experimental software. A solution is needed to allow experimenters in vision to integrate stimulus modification routines within general experimental display software. The UCSB Psychophysics Toolbox (Brainard, 1997) can run image-processing and psychophysical investigations on the Macintosh (in tandem with VideoToolbox [Pelli, 1997] if desired) with MATLAB. VideoToolbox and Psychophysics Toolbox are free, but MATLAB is quite expensive.[2] MATLAB involves manipulations of matrices, which is advantageous for periodic stimuli (such as gratings) but is perhaps confusing for the less mathematically sophisticated. Getting started with its command line interface requires more learning time than does menu-driven software.

This research was supported by BBSRC Grant BB039AP50 to T.R.J. We are grateful to Paul Sergeant for his advice on Cogimatic and Delphi and to Axel Monteiro for his help with proofing and implementing the batch listing. Correspondence concerning this article should be addressed to K. C. Scott-Brown, School of Psychology, University Park, Nottingham, NG7 2RD England (e-mail: ksb@psychology.nottingham.ac.uk).

The modern "Pentium-based" PC platform, ideally suited to these applications, is less well served with suitable image analysis software for quick and easy implementation of experimental runs of stimuli.[3] One exception is the IPRS image-processing and pattern recognition system (Caelli, Dillon, Osman, & Krieger, 1997) for the PC or SGI and Suns (under X-Windows). Unfortunately, it requires the Linux operating system on the PC, which, although free, has a much smaller user base than does Windows. Table 1 lists some of the key image-processing packages currently available as well as several IDEs.

THE VISION STARTER KIT

In this article, we introduce a budget image-processing solution for the Windows 3.1/95/98/NT PC platform. Cogimatic's Vision Starter Kit (VSK) enables the development of tailor-made applications that precisely define transformations to digitized images and that automate the production and development of the stimuli into convenient batches. The system is based around a dynamic linked library (DLL), and its interface source code contains many standard algorithms used in vision research. A set of wrapper classes in C++, Visual Basic, and Delphi means that the programming requirements of the user are limited to the sequencing of operations rather than their low-level implementation. We chose Delphi as our programming environment.[4]

AUTOMATIC FILTERING OF A SERIES OF IMAGE FILES

The demonstration project supplied with VSK is similar to NIH Image and is a menu-driven image analysis program allowing the user to open files and apply image transformations and filters. The user can thus immediately start image processing with the basic DLL functions. However, instead of having macro functionality like that of NIH Image, the user can directly program the VSK DLL and maximize the use of the library of routines either by making some modifications to the demonstration project or by writing completely new software to allow fully flexible automatic image processing. For example, in our own experiments (Jordan & Scott-Brown, 1999, 2000) we use tachistoscopic presentation of low spatial frequency filtered words and nonwords contained in bitmap files. The listing in Appendix B shows a simple addition to the VSK's grayscale demo project that can be used to automatically filter a series of such files. Complex sequences of image manipulations (including 8- and 32-bit fast Fourier transforms) are achievable, but for clarity we restrict this routine to relatively straightforward operations. Our series of 256×256 bitmap files contained black and white images with a 256 gray-level range. The batch program takes a list of the filenames from a textfile and opens up each bitmap file in turn and filters its contents. The image is first made negative, then rotated through 90°, then mirror reversed, and finally filtered with a 3×3 mean filter. The resultant image is saved to a new file with the number "1" appended to the original bitmap file name. Implementing new filters requires simple parameter changes or function calls. To produce a new run of stimuli without the batch program would require up to 3 days of work and would run the risk of introducing accidental errors.

A nondigital, diffuser-based system is not as effective because of the bulky and expensive nature of the materials, the complexity and cost of the mechanical handling,

Table 1
Platforms and Prices for Image Processing Packages and Software Development Suites

Application	Platform	Publisher's Price ($)
Vision Starter Kit (Version 2)	PC (Win95/98/NT4)	49.95
NIH Image	Mac	Free
Image J	Platform indep. (Java)	Free
Scion Image	PC	Free
Photoshop	Mac/PC	499.00
PhotoScripter	Mac	299.00
Image Pro Plus	PC	1,500.00
MATLAB	PC (& Mac to Version 5.2)	499.00
MATLAB Signal Processing Toolbox	PC	199.00
Delphi Standard	PC	99.95
Delphi Professional	PC	799.00
Microsoft C++	PC	99.95
Borland C++ Builder	PC	99.95
Visual Basic Professional 6.0	PC	422.00
Visual Basic (Academic)	PC	99.95

This table is in no way a complete listing of all available products but merely illustrates the differences between the products at the time of writing. Prices are in U.S. dollars. Some of these products (particularly the IDEs) are subject to significant academic discount rates. There may be restrictions or extensions on these rates depending on the affiliation of your particular institution. Appendix A lists contact details for products.

the risk of corruption of the experiments due to the lags and mechanical noise in filter changes, and the cost of implementing parameter changes.

Once installed, the system permits the freedom to rapidly explore the development of stimulus sets. Highly complex image processing is performed in exactly the same way as are these examples, and the variety of visual stimuli that can be used is vast. We used text but could equally well use faces, natural scenes, fractals, or any other digitized image. If preferred, the VSK can produce hard copy with a good quality laser printer (e.g., for development of acuity-card style tests).

Users should test the output of any of their programs and the original filters to confirm that they are performing as expected. The simplest method is to compare operations on sample stimuli (e.g., gratings) with the output in a different package such as Image-J (the Java version of NIH Image), which can run off an internet browser. We compared the output of the listing in this article with the equivalent operations in NIH Image and Image-J, using Photoshop to inspect the gray values for each pixel. The postfiltering gray levels were identical in each package.

The power behind this system is the DLL, which allows programmers to take advantage of fully integrated routines. The latest generations of Pentium II and III level processors are fast enough to permit substantial amounts of image analysis in a short time. Consequently, more of an experiment can be run on line, which can dramatically save disk space (only originals need be stored), as well as improve the portability of the experimental suite. In psychophysical investigations, which can involve thousands of trials, the economies of scale that accrue are considerable. This is particularly relevant for response-dependent experimental paradigms, such as staircase procedures, which can involve hundreds of fractional increments and decrements in a given stimulus parameter to determine a threshold. Storing all permutations is unattractive, owing to the numbers of files involved, but a dynamic modification system neatly sidesteps this problem and brings a significant advantage over stand-alone packages. The DLL provides a major advantage over some other solutions because it can be included in external projects and disseminated without a new license being paid; only the software kit itself requires the license.

Using a DLL to store the routines brings all the advantages of an integrated software package but retains flexibility, allowing the software upgrades to grow with hardware requirements. Larger projects become difficult to manage in a macro language environment (the maximum file size in NIH Image is 32K); both Delphi and C++ are fully object oriented, however, allowing modular software management and development. As projects grow, object-oriented development effectively reduces complexity and expedites coding efficiency. Other systems become cumbersome and the code becomes less reusable, particularly when new personnel become involved partway through a project. Of course the user is not required to code in an "object-oriented" manner (linear programs will still work); but the functionality afforded by these IDEs offers huge advantages over macro languages.

Version 2 of the VSK is now available and includes additional color (RGB) source code and demonstration software as well as the original grayscale functionality of Version 1.22. The new 32-bit implementation allows the algorithms to run faster.[5]

In conclusion, the Macintosh is relatively well served with image manipulation software for perceptual and cognitive research, but equivalent applications for the PC are limited in availability. We have presented a system for the PC that allows powerful image processing, but with the added computational power of being fully programmable. The use of Delphi or Visual Basic allows less than expert coders to produce working programs much more quickly than in a macro language, thanks to the integrated debugging facilities. The DLL allows imaginative programmers to fully integrate image-processing techniques into their software in an easily maintainable and upgradable fashion. The result is a low-cost, readily available, fast and flexible system, which is applicable on a broad platform of PC hardware. Experimenters can precisely specify parameters of stimulus manipulation and run the filtering operations in large batches. Alternatively, they can combine the image transformations with experimental software to fully harness the power and flexibility of the routines.

REFERENCES

Alexander, K. R., Xie, W., & Derlacki, D. J. (1994). Spatial-frequency characteristics of letter identification. *Journal of the Optical Society of America A*, **11**, 2375-2382.

Braddick, O., Campbell, F. W., & Atkinson, J. (1978). Channels in vision: Basic aspects. In R. Held, H. W. Leibowitz, & H.-L. Teuber (Eds.), *Handbook of sensory physiology: Vol. 8: Perception* (pp. 3-38). New York: Springer-Verlag.

Brady, N., Bex, P. J., & Fredericksen, R. E. (1997). Independent coding across spatial scales in moving fractal images. *Vision Research*, **37**, 1873-1883.

Brainard, D. H. (1997). The psychophysics toolbox. *Spatial Vision*, **10**, 433-436.

Caelli, T., Dillon, C., Osman, E., & Krieger, G. (1997). The IPRS image processing and pattern recognition system. *Spatial Vision*, **11**, 107-115.

Campbell, F. W., & Robson, J. G. (1968). Application of Fourier analysis to the visibility of gratings. *Journal of Physiology*, **197**, 551-566.

Fiorentini, A., Maffei, L., & Sandini, G. (1983). The role of high spatial frequencies in face perception. *Perception*, **12**, 195-201.

Graham, N. V. S. (1989). *Visual pattern analyzers*. New York: Oxford University Press.

Hogben, J. H., Pratt, C., Dedman, K., & Clark, C. D. (1996). Blurring the image does not help disabled readers. *Vision Research*, **36**, 1503-1507.

Jordan, T. R., & Scott-Brown, K. S. (1999). *Word superiority effects with visually filtered strings: Evidence for coarse visual cues in word recognition*. Paper presented at the July meeting of the Experimental Psychology Society, University of Durham.

Jordan, T. R., & Scott-Brown, K. S. (2000). *Word superiority effects with filtered strings: Evidence for coarse visual cues in word recognition*. Manuscript in preparation.

LANDY, M. S., COHEN, Y., & SPERLING, G. (1984a). HIPS: Image processing under UNIX. Software and applications. *Behavior Research Methods, Instruments, & Computers*, **16**, 199-216.

LANDY, M. S., COHEN, Y., & SPERLING, G. (1984b). HIPS: A Unix-based image processing system. *Computer Vision, Graphics, & Image Processing*, **25**, 331-347.

LEGGE, G. E., PELLI, D. G., RUBIN, G. S., & SCHLESKE, M. M. (1985). Psychopyhsics of reading: 1. Normal vision. *Vision Research*, **25**, 239-252.

MAY, J. G., BROWN, J. M., SCOTT, S., & DONLON, M. (1990). Visual persistence of spatially filtered images. *Perception & Psychophysics*, **47**, 563-567.

OLDS, E. S., & ENGEL, S. A. (1998). Linearity across spatial frequency in object recognition. *Vision Research*, **38**, 2109-2118.

PARISH, D. H., & SPERLING, G. (1991). Object spatial frequencies, retinal spatial frequencies, noise, and the efficiency of letter discrimination. *Vision Research*, **31**, 1399-1415.

PARKER, D. M., LISHMAN, J. R., & HUGHES, J. (1992). Temporal integration of spatially filtered images. *Perception*, **21**, 147-160.

PELLI, D. G. (1997). The VideoToolbox software for visual psychophysics: Transforming numbers into movies. *Spatial Vision*, **10**, 437-442.

SCHYNS, P. G., & OLIVA, A. (1997). Flexible, diagnosticity driven, rather than fixed, perceptually determined scale selection in scene and face recognition. *Perception*, **26**, 1027-1038.

SOLOMON, J., & PELLI, D. (1994). The visual filter mediating letter identification. *Nature*, **369**, 395-397.

WATT, R. J. (1991). *Understanding vision*. London: Academic Press.

NOTES

1. Applescript can help automate work in Photoshop but only in a limited fashion because of the limited dictionary available in Photoshop. The purchase of Photoscripter 1.0 solves many of these problems but at an additional cost of $299 on top of the price of Photoshop.

2. MATLAB is only available up to Version 5.2 on the Macintosh platform. Unfortunately the Psychophysics Toolbox is not implemented for PC versions of MATLAB (the PC signal-processing unit may be of use for a further $200, however). The long-term risk in running MATLAB on the Mac is that future operating systems will not run MATLAB 5.2. MATLAB may provide the backbone for a powerful image analysis suite for a laboratory. Our own proposals to use a Pascal-based system has advantages for some users. Individuals are much more likely to come across Pascal in high school, and those exposed to computer science courses will almost certainly work in C or C++ along with object-oriented programming at some point. Pascal coders need only basic knowledge of the difference between integers and floating point numbers to produce working code.

3. Windows users can use a version of NIH Image written in Java: Image-J can be run off one's browser as an applet (http://rsb.info.nih.gov/ij/). Scion Image has written a beta version of NIH image for Windows (http://www.scioncorp.com/pages/scion_image_windows.htm).

4. Delphi is a Pascal-based object-oriented Windows development tool. The Delphi language is essentially Pascal, with some proprietary syntax to integrate the extensive user interface and internal event handling routines that form the backbone of the package.

5. Version 1.22 of the Vision Starter Kit is a 16-bit system, whereas Version 2 requires the 32-bit versions of Delphi (Versions 3, 4, and 5). Version 1.22 will work with Windows 3.1.

APPENDIX A

Information on The Vision Starter Kit software can be found on the Internet at www.cogimatic.com. Cogimatic can be reached by mail, phone, or e-mail: Cogimatic, P.O. Box 471678, San Francisco, CA 94147; (415) 922-7662; contact@cogimatic.com. Version 2 of the kit is available from DigiBuy over the Internet, fax, or phone: http://www.digibuy.com/cgi-bin/order.html?cogimatic+95662927414. Version 1.22 of the kit is still available from Software.Net (part of the Beyond.Com company): http://www.software.net/PKSN080140/prod.htm.

NIH Image was developed at the U.S. National Institutes of Health and is available on the Internet at http://rsb.info.nih.gov/nih-image/.

Scion Image is available from Scion Corporation, 82 Worman's Mill Court, Suite H, Frederick, MD 21701: http://www.scioncorp.com/frames/fr_scion_products.htm.

Adobe Photoshop is available from Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110-2704: http://www.adobe.com/products/photoshop/main.htm.

PhotoScripter is available from Main Event, PO Box 21470, Washington, DC 20009: http://www.mainevent.com/photo.html.

Image Pro Plus is available from Media Cybernetics, 8484 Georgia Ave., Silver Spring, MD 20910: http://www. mediacy.com/ippage.htm.

MATLAB is available from The MathWorks, 24 Prime Park Way, Natick, MA 01760-2098: http://www.mathworks.com/.

Delphi and C++ Builder are available from Borland, Inprise Corporation, Worldwide Headquarters, 100 Enterprise Way, Scotts Valley, CA 95066: http://www.borland.com/delphi/ & http://www.borland.com/bcppbuilder/.

Microsoft C++ and Visual Basic are available from Microsoft, One Microsoft Way, Redmond, WA 98052-6399: http://msdn.microsoft.com/vbasic/ & http://msdn.microsoft.com/visualc/.

(*Continued on next page*)

APPENDIX B

The example below is a method of loading up a sequence of image files and performing a series of image operations and transformations before saving the results in new files. It aims to show the ease of implementation of the routines in software and additionally how the routines can be used dynamically and automatically in a program. Comments are provided to allow the reader to follow the flow. The kit itself comes with an electronic manual describing all the functions available in the DLL. Please note that the VSK is required before this listing can be successfully incorporated into a working program. This routine is written for Delphi and can be pasted into the *grayscale* demo project supplied with the VSK. Delphi projects come with easily modified graphical user interfaces, and the demo package is no exception; the addition of a button on its user interface form will provide the three lines of code (indicated by {++} in the listing). One should ensure that they immediately precede the last three lines in the demo.pas file. One simply inserts the rest of the listing in the appropriate spaces.

Listing
A Sample Routine for Filtering a Series of Image Files

```
procedure TMainForm.Button1Click(Sender: TObject);                   {++}  {+Delphi creates this line+}
var
 IsOkay : Boolean;
 BatchList : Tstrings;                                               {Variables to handle files in the list}
 MyFileName : string;
 Base, DestBase, StimFile, StimOutFile : string;
 StimCount : integer;                                                {Keeps track of the loop}

procedure RunBlur (OrigFile : string);                               {RunBlur Procedure: used in each loop iteration}

begin
    MyFilename := OrigFile;                                          {Assign a variable for the filename passed to
                                                                     this procedure}
    ImageResult.Free ;                                               {Release Current Memory Buffer with 'Free'}

    ImageResult:=Gr8Image.CreateFromDIBFile( MyFilename, StdWts ) ;  {'Create' acquires a new appropriately sized
                                                                     buffer before loading the data. The file must be
                                                                     in the Windows DIB(Device Independent
                                                                     Bitmap) format (usually as *.BMP or *.DIB).}
    PaintBox1.Invalidate ;                                           {The 'PaintBox' is a simple graphic control (i.e.,
                                                                     a Delphi Object) that provides a canvas for ap-
                                                                     plications to use for rendering an image. 'Invali-
                                                                     date' tells Windows to repaint the control imme-
                                                                     diately after other important Windows messages
                                                                     are handled.}
    Screen.Cursor := crHourglass;                                    {Indicate Busy State on Mouse Pointer}

    isOkay := ImageResult.Negative ;                                 {Take the Negative of the Image}
      PaintBox1.Invalidate ;
      if not isOkay then
        MessageDlg('Negative Error.', mtWarning, [mbOK], 0);         {Flag the user with a dialog box if it fails}

    isOkay := ImageResult.Rotate90(1) ;                              {Rotate the Image}
      PaintBox1.Invalidate ;
      if not isOkay then
        MessageDlg('Rotation Error.', mtWarning, [mbOK], 0);

    isOkay := ImageResult.FlipX( );                                  {Mirror reverse the Stimulus}
      PaintBox1.Invalidate ;
      if not isOkay then
        MessageDlg('Clipping Error.', mtWarning, [mbOK], 0);
    isOkay := ImageResult.FastMeanFilter(3, 3) ;                     {Filter the Image: filter size defined by
                                                                     parameters}
      PaintBox1.Invalidate ;
      if not isOkay then
        MessageDlg('Filter Error.', mtWarning, [mbOK], 0);

      Screen.Cursor := crDefault ;                                   {Change the pointer back from an HourGlass}
end;                                                                 {end of RunBlur procedure}
```

APPENDIX B (Continued)

```
begin                                                          {++}  {+Delphi creates this line+}
    Button1.Enabled := False;                                  {Disable the Button during operation}
    Button1.Caption := 'Running';
    BatchList := TStringList.Create;                           {Create a string list 'object' to store and manipu-
                                                               late the list of strings}
    Base := 'A:\';                                             {change this to refer to your own source direc-
                                                               tory if necessary}
    DestBase := 'A:\';                                         {defines a destination directory for the trans-
                                                               formed files}
    BatchList.LoadFromFile('A:\textfile.txt');                 {List your bitmap files in a textfile} {Note:
                                                               Each filename should appear on a separate line
                                                               without the .bmp suffix. A carriage return
                                                               should immediately follow each name.}
    for StimCount := 0 to 3 do                                 {Where the second number is the number of
                                                               files to filter minus 1}
      begin
        StimFile := Base + BatchList[StimCount] + '.bmp';      {Select the image file}
        StimOutFile := DestBase + BatchList[StimCount] +'1' + '.bmp';  {Defines the output file: Directory path, stimu-
                                                               lus name, the suffix '1' and the .bmp suffix}
        RunBlur(StimFile);                                     {Perform the image transformations}
        SaveToFile(ImageResult, StimOutFile);
        Refresh;                                               {Clear the Picture Display}
      end;
    BatchList.Free;                                            {Free up memory}
    Button1.Caption := 'Done';
end;                                                           {++}  {+Delphi creates this line+}

                  {=======End of 'A sample routine for filtering a series of image files'=======}

initialization                                                 {initialization may appear as ' begin'}
    initConversionTable;
end.                                                           {These three (pre-existing) lines must remain at
                                                               the end of the demo.pas file}
```

(Manuscript received May 15, 1999;
revision accepted for publication May 15, 2000.)