Behavior Research Methods, Instruments, & Computers
1986, 18 (1), 1-9

METHODS & DESIGNS

Perceptual identification of visually degraded stimuli

JOHN R. VOKEY and JOHN G. BAKER
University of Lethbridge, Lethbridge, Alberta, Canada

GORDON HAYMAN
University of Toronto, Toronto, Ontario, Canada

and

LARRY L. JACOBY
McMaster University, Hamilton, Ontario, Canada

In this article, we describe procedures, materials, and some representative results of a microcomputer-based approach to the degradation of visual stimuli for the investigation of perceptual identification. We discuss application of the procedures for the production of visually degraded picture, letter, and word stimuli, and of visual stimuli common to neuropsychological investigations.

The use of visually degraded stimuli for the investigation of perceptual processes has a long history within psychology. Typically, the intent of this approach has been to slow down the processes, making more readily observable the subprocesses, subcomponents, and time course of visual perception that often are masked by the rapidity and automaticity of normal visual perception. The most common technique probably is the visual degrading of a stimulus through the use of brief exposure durations (e.g., Sperling, 1960), typically via a tachistoscope or, more recently, computer emulations thereof. Other common methods include the blurring of the target (Bruner & Potter, 1964) and the masking of one stimulus by the superimposition of or replacement with another.

In this article, we present the materials, methods, and some representative results of another approach to the visual degradation of stimuli that is implemented on an Apple II (II+, IIe, IIc) microcomputer or a clone. The approach is similar to the signal-detection theoretic approach to perception, wherein the object of (at least part of) the perceptual identification system is seen as the disambiguation of stimulus signal from an overlay of both endogenous and exogenous noise (Green & Swets, 1966). The procedures we describe allow the experimenter to exercise precise control over the degree of exogenous noise added to a stimulus, and provide a relatively simple method for the investigation of factors related to the perceptual identification of visual stimuli. The procedures are easily generalized to produce visual displays similar to those used in neuropsychological test batteries and in investigations of the consequences of various forms of brain damage on perceptual identification (e.g., Warrington, 1982).

THE TASKS

Mask Clarification

The basic procedure begins with the visual display of a picture overlaid with a random noise mask on the computer video display. Initially, the picture is completely masked by the noise, but over trials, the ratio in the display of pixels (picture elements) emanating from the picture to those emanating from the noise slowly increases until the subject can correctly identify (name) the picture (see Figure 1). In most of our investigations, clarification trials have been subject-paced; the subject clarifies the picture a step at a time by pressing a key on the computer keyboard, stopping when he or she can correctly identify the picture. This procedure results in the simple dependent measure of number of keypresses (or percentage clarified) to correct identification, although total time taken to clarify the picture to the point of correct identification also may be recorded (Brooks, Jacoby, & Whittlesea, in preparation). Figure 1 displays an abbreviated sequence of the mask clarification of a picture of an elephant.

The research reported in this article was supported by a grant from the Natural Sciences and Engineering Research Council of Canada to the first author. Requests for reprints should be sent to J. R. Vokey, Department of Psychology, University of Lethbridge, Lethbridge, Alberta, Canada T1K 3M4.

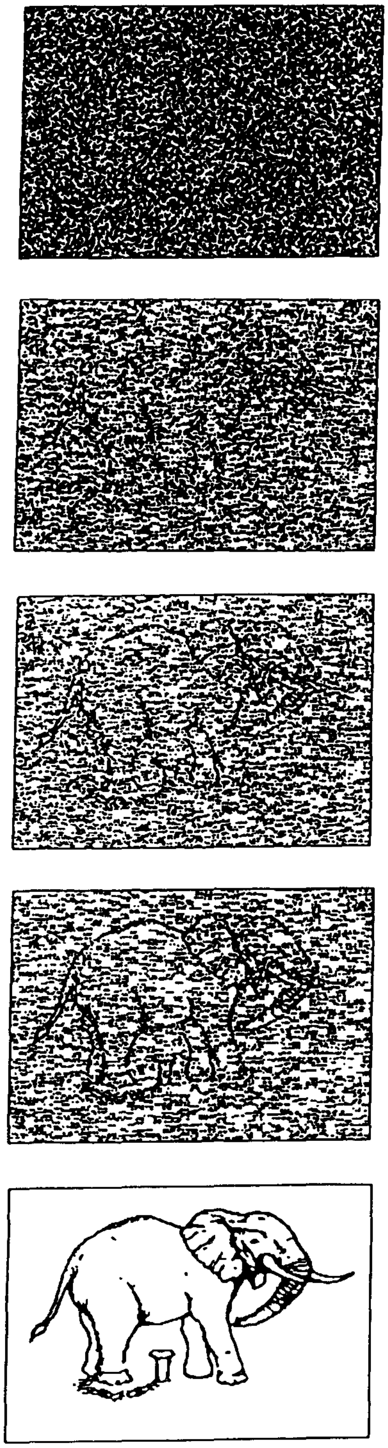

Figure 1. An abbreviated example of the sequence of stimuli produced by the mask-clarification procedure. Reading from top to bottom, the percentage clarified is 0%, 30%, 45%, 60%, and 100%, respectively.

In a representative experiment using the mask-clarification procedure on the effect of a single prior exposure on perceptual identification, each of 5 subjects was exposed to a different random set of 20 fully clarified pictures at a rate of 6 sec per picture. Following this initial exposure, each subject was presented with the clarification task for a randomly ordered set of 30 pictures. For each subject, one third of the pictures were *identical* to 10 pictures randomly chosen from those previously exposed. Another one third of the pictures differed from those in the preexposure set, but had the *same name* as the remaining 10 preexposed pictures. The remaining one third of the pictures were new and, hence, had *different* names than did the pictures in the preexposure set. There was a large effect of this variation in picture type on the percentage clarified for correct identification [$F(2,8) = 65.26$, $MSe = 1.59$, $p < .0001$]. Subjects required clarifications of 33.9%, 38.1%, and 43% to identify *identical*, *same name*, and *different* pictures, respectively. Although prior exposure of a picture's name ("priming") can be seen to assist perceptual identification (i.e., *same name* pictures required less clarification than did *different* pictures), it is clear from the results that a single prior exposure to a picture can enhance perceptual identification to a level beyond that of simple name priming. The theoretical consequences of these and similar results for notions such as Warren and Morton's (1982) "pictogen" model of perceptual identification, as well as the large role played by specific familiarity, are discussed in Jacoby and Brooks (1984) and Brooks, Jacoby, and Whittlesea (in preparation).

Dot Clarification

A minor change in the computer routines (discussed below) used in the mask-clarification procedure yielded a similar task which we call dot clarification. In this task, each picture appears initially as a blank display. Over trials (again, each trial is typically initiated by the subject's pressing a key), the picture is slowly built up as randomly chosen pixels of the picture are illuminated, producing a series of stimuli similar to Gollin's (1960) incomplete pictures. Figure 2 shows a picture of an elephant taken through an abbreviated sequence of dot clarification. The pictures of elephants in Figures 1 and 2 also provide an example of the picture pairs used to generate the *identical* and *same name* stimuli discussed earlier.

With the exception of the change in the clarification task, the next experiment was a replication of the earlier experiment with mask clarification. Each of 5 subjects was presented with a different random set of 20 fully clarified pictures at an exposure rate of 6 sec per picture, followed by the dot clarification task for 10 *identical*, 10 *same name*, and 10 *different* pictures. As in the previous experiment, there was a large effect of picture type [$F(2,8) = 53.45$, $MSe = 4.79$, $p < .0001$]. Again, *identical* pictures were identified with less clarification (10%) than were *same name* pictures (20.6%) and *different* pictures required the greatest degree of clarification (23.6%). Moreover, comparison of the results of the two clarification procedures indicates not only that dot-clarified pictures required less clarification than did mask-clarified pictures before being correctly identified [$F(1,8) = 160.13$, $MSe = 19.27$, $p < .0001$], but that the effect of picture type was significantly larger for dot clarifica-

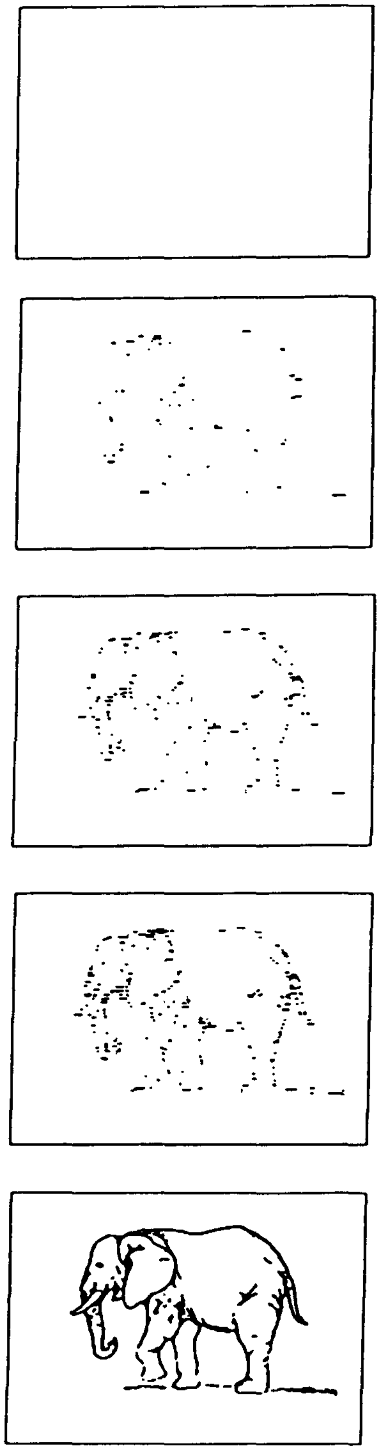

Figure 2. An abbreviated example of the sequence of stimuli produced by the dot clarification procedure. Reading from top to bottom, the percentage clarified is 0%, 10%, 20%, 30%, and 100%, respectively.

tion than it was for mask clarification [$F(2,16) = 8.54$, $MSe = 3.19$, $p < .0030$]. This increase in the slope of the function relating percentage clarified to picture type suggests that in the absence of exogenous noise, the relative advantage in perceptual identification for previously exposed pictures is enhanced.

DESCRIPTION OF THE SOFTWARE

Construction and Storage of Pictures

Although virtually any pictures may be used, the bulk of our high-resolution pictures have been simple line drawings and line-shot photographs taken from such sources as the Peabody picture vocabulary (Dunn, 1965), the Mooney picture set (Mooney, 1956, 1957), the Snodgrass and Vanderwart (1980) picture set, and children's coloring books. Each picture is represented digitally by a video digitizer (Dithertizer II, Computer Stations, Inc., 1980) and stored to disk.

On the Apple II, each high-resolution picture occupies 32 pages (8,192 bytes) of memory, which translates into 34 sectors (Apple's DOS 3.3) or 17 blocks (Apple's ProDOS) when stored to disk. This file space required for the pictures permits an upper limit of only 14 pictures per diskette; however, the two experiments described above, for example, required a minimum of 60 pictures (30 *same name* pairs) to be on-line simultaneously. To circumvent the limited capacity, the simple data-compression algorithm, called KRUNCH (shown in Listing 1), was developed. For simple line drawings, such as those shown in Figures 1 and 2, it is possible to store more than 60 pictures to a single diskette by using KRUNCH. The algorithm, written in 6502 assembler language, assumes that the picture to be compressed is a simple white on black line drawing residing on the second high-resolution graphics page of the Apple II. When called, the routine scans the picture, storing the location and value of every nonzero (i.e., nonblack) byte encountered to Page 1 of high-resolution graphics; from there, the resulting compressed data may be saved to disk with the command: BSAVE PICNAME, A$2000, LPEEK(249) + PEEK(250) * 256 − 8191. To display a compressed picture, the file is loaded from the disk to high-resolution graphics Page 1, and then the assembler language routine, called UNKRUNCH (also shown in Listing 1) is called to recreate the original picture on high-resolution graphics Page 2.

Construction of the Random Mask

Both the mask- and dot-clarification procedures use an 8192-byte sequence of random 8-bit values to control the clarification of the visual display. For convenience, these values are stored as a pseudo high-resolution picture immediately above the memory locations reserved by the Apple II for high-resolution graphics Pages 1 and 2, beginning at address 24576 ($6000). Construction of the mask consists of storing a random sequence of the values between 0 and 255 ($0 - $FF) into the appropriate memory locations. Using Applesoft BASIC's pseudorandom number function,[1] the mask may be constructed by executing the BASIC statement, FOR I = 24576 TO 24576+8191: POKE I, RND(1)*256: NEXT I, and then saved to disk with the command BSAVE MASK, A$6000, L$2000.

Listing 1

```
0000:                 **********************************
0000:                 *                                *
0000:                 *      PICTURE COMPRESSION       *
0000:                 *                                *
0000:                 **********************************
0000:                 ;
0000:                 ; Copyright (c) 1985
0000:                 ; John R. Vokey & John G. Baker
0000:                 ;
0000:                 *================================
0000:                 ;               EQUATES
0000:                 *================================
0000:                 ;
0000:         0020    HPAG1     EQU $20
0000:         0040    HPAG2     EQU $40
0000:         0060    HPAG3     EQU $60
0000:         00F9    PAG1      EQU $F9
0000:         00FB    PAG2      EQU PAG1+2
0000:                 ;
0000:                 *================================
0000:                 ;               KRUNCH
0000:                 *================================
0000:                 ;
0300:         0300              ORG $300          CALL 768 from BASIC
0300:                 ;
0300:A9 40            KRUNCH    LDA #HPAG2        original on HPAG2
0302:85 FC                      STA PAG2+1
0304:A9 20                      LDA #HPAG1        result to HPAG1
0306:85 FA                      STA PAG1+1
0308:A0 00                      LDY #0
030A:84 FB                      STY PAG2
030C:84 F9                      STY PAG1
030E:                 ;
030E:B1 FB            LOOP1     LDA (PAG2),Y      get picture byte
0310:29 7F                      AND #%01111111    Strip colour bit (if set)
0312:F0 1A    032E              BEQ NEXTY         If zero, go
0314:98                         TYA               Else, save current Y
0315:48                         PHA               on stack
0316:B1 FB                      LDA (PAG2),Y      Recover byte
0318:A0 00                      LDY #0            set Y=0
031A:91 F9                      STA (PAG1),Y      Save to page 1
031C:C8                         INY               next byte
031D:68                         PLA
031E:91 F9                      STA (PAG1),Y      save Y of pic byte
0320:A8                         TAY               advance page 1 pointer
0321:A5 F9                      LDA PAG1
0323:18                         CLC
0324:69 02                      ADC #2
0326:85 F9                      STA PAG1
0328:A5 FA                      LDA PAG1+1
032A:69 00                      ADC #0
032C:85 FA                      STA PAG1+1
032E:                 ;
032E:C8               NEXTY     INY               next pic byte
032F:D0 DD    030E              BNE LOOP1         If more on this line, go
0331:98                         TYA               Else,
0332:91 F9                      STA (PAG1),Y      mark end-of-line
0334:E6 F9                      INC PAG1          Bump page 1 pointer
0336:D0 02    033A              BNE NXTLIN
0338:E6 FA                      INC PAG1+1
033A:                 ;
033A:E6 FC            NXTLIN    INC PAG2+1        Next line
033C:A5 FC                      LDA PAG2+1
033E:C9 60                      CMP #HPAG3        Done?
0340:90 CC    030E              BCC LOOP1         No, go again
0342:60                         RTS               Yes, return to caller
0343:                 ;
0343:                 *================================
0343:                 ;              UNKRUNCH
0343:                 *================================
0343:                 ;
0300:         0300              ORG $300          CALL 768 from BASIC
0300:                 ;
0300:A9 20            UNKRUNCH  LDA #HPAG1        Compressed pic on page 1
0302:85 FA                      STA PAG1+1
0304:A9 40                      LDA #HPAG2        Result to page 2
0306:85 FC                      STA PAG2+1
0308:A0 00                      LDY #0
030A:84 F9                      STY PAG1
030C:84 FB                      STY PAG2
030E:                 ;
030E:B1 F9            DRAW      LDA (PAG1),Y      recover compressed byte
0310:F0 19    032B              BEQ NXTLIN2       If zero, next line
0312:48                         PHA               Else, save byte on stack
0313:E6 F9                      INC PAG1          Point to next byte
0315:D0 02    0319              BNE DRAW1
0317:E6 FA                      INC PAG1+1
```

Listing 1, continued

```
0319:B1 F9           DRAW1    LDA (PAG1),Y    Recover old Y
031B:A8                       TAY
031C:68                       PLA             Recover pic byte
031D:11 FB                    ORA (PAG2),Y    OVERLAY it
031F:91 FB                    STA (PAG2),Y    and store to page 2
0321:                ;
0321:A0 00           NXTBYT   LDY #0          set Y=0
0323:E6 F9                    INC PAG1        bump page 1 pointer
0325:D0 E7    030E            BNE DRAW        and go again
0327:E6 FA                    INC PAG1+1
0329:D0 E3    030E            BNE DRAW        always taken
032B:                ;
032B:E6 FC           NXTLIN2  INC PAG2+1      next line
032D:A5 FC                    LDA PAG2+1
032F:C9 60                    CMP #HPAG3      Done?
0331:90 EE    0321            BCC NXTBYT      No, go again
0333:60                       RTS             Yes, return to caller
```

Mask- and Dot-Clarification Procedures

The heart of the clarification procedures is contained in the assembler language routine shown in Listing 2. It is written to be coresident in a typically unused area of memory (Page 3) with the routine that is used to recreate compressed pictures, so that both will be available for use from within a controlling Applesoft BASIC program. Both mask and dot clarification are handled by the same routine; which procedure is executed on a given call to the routine is determined by the setting of two bytes, which are passed (using the BASIC POKE command) to the routine from BASIC. POKE 850, 176: POKE 851, 253 sets mask clarification, and POKE 850, 169: POKE 851, 0 sets dot clarification. The amount of clarification for a given call to the routine is determined by the value of another byte, called STEP, which is similarly passed to the routine (POKE 255, STEP) from the controlling BASIC program.

For both mask and dot clarification, the picture to be clarified resides on high-resolution graphics Page 2. On the Apple II, each byte of the high-resolution graphics page controls the display (on/off) of seven horizontally consecutive pixels. (The eighth bit of each byte, which is cleared by the clarification routines, normally controls the color of the pixels in the byte.) Each time the clarification routine is called, it cycles through each of the 8192 bytes of the picture, and the result of the process is stored to high-resolution graphics Page 1, where it is displayed. For each picture byte, the routine compares the value of a byte from the same relative position in the random mask to the value of the STEP byte. If the random value exceeds that of the STEP byte, then the random byte (for mask clarification) or a zero (for dot clarification) is transferred to the display page. Otherwise, the picture byte is transferred and displayed. Because clarification occurs by swapping picture and mask bytes, masking is normally limited to a minimum of 7 pixels, although modifying the mask between calls to the routine will allow individual pixels to be masked. By successively incrementing the STEP value between calls to the routine, a picture may be taken through 256 different levels of clarification. More rapid rates of clarification may be achieved by using larger increments of the STEP value between calls to the routine. A STEP value of 255 will result in a fully clarified copy of the picture on graphics Page 2 being transferred to the display page.

Inversion and Reflection Routines

Also included in Listing 2 are two further routines designed as examples of the types of manipulations that may be performed on the high-resolution image before it is transferred to the display page. The first of these, called INVERT, is used to complement the image residing on high-resolution graphics Page 2. Calling the routine will invert a white image on a black background, for example, to a black image on a white background. Calling the routine again, will invert the image back to its original form. The routine has other uses as well; by pointing it at the random mask rather than at the picture, for example, and passing it a random value to be used in the exclusive-or (EOR) operation, the INVERT routine provides a rapid method of randomizing the mask between different pictures.

The second routine, called REFLECT, performs a mirror-image (left-to-right) transformation of the high-resolution picture. As with INVERT, the process is completely reversible; calling the routine twice in succession will first reflect the image, then reflect it back to its original form.

Overlaying Pictures

The UNKRUNCH routine may be used to do more than recreate previously compressed high-resolution displays. In particular, it was constructed to overlay the picture being recreated on whatever is currently residing on the high-resolution graphics page. Typically, the desired background is a blank display produced by calling Applesoft BASIC's clear high-resolution routine (i.e., POKE 230, 64: CALL 62450), but it need not be. Calling the UNKRUNCH routine without first clearing high-resolution graphics Page 2 will result in the recreated picture's being merged with the current image. Thus, for example, stimuli such as the overlapping figures used by Ghent (1956) may easily be created (see Figure 3). Simi-

Listing 2

```
0000:                ********************************
0000:                *                              *
0000:                *     PICTURE CLARIFICATION    *
0000:                *           ROUTINES           *
0000:                *                              *
0000:                ********************************
0000:                ;
0000:                ; Copyright (c) 1985
0000:                ; John R. Vokey and John G. Baker
0000:                ;
0000:                ; These routines are written to be
0000:                ; co-resident with the UNKRUNCH routine
0000:                ;
0000:                *================================
0000:                ;               EQUATES
0000:                *================================
0000:                ;
0000:          0020  HPAG1      EQU   $20
0000:          0040  HPAG2      EQU   $40
0000:          0060  HPAG3      EQU   $60
0000:          00F9  PAG1       EQU   $F9
0000:          00FB  PAG2       EQU   PAG1+2
0000:          00FD  PAG3       EQU   PAG2+2
0000:          00FF  STEP       EQU   PAG3+2
0000:                ;
0000:                *================================
0000:                ;               CLARIFY
0000:                *================================
0000:                ;
0334:          0334             ORG   $334           CALL 820 from BASIC
0334:          0000  DOT        EQU   0              conditional assembly
0334:                ;
0334:A9 20           CLARIFY    LDA   #HPAG1         Display on page 1
0336:85 FA                      STA   PAG1+1
0338:A9 40                      LDA   #HPAG2         Pic on page 2
033A:85 FC                      STA   PAG2+1
033C:A9 60                      LDA   #HPAG3         Random mask on page 3
033E:85 FE                      STA   PAG3+1
0340:A0 00                      LDY   #0
0342:84 F9                      STY   PAG1
0344:84 FB                      STY   PAG2
0346:84 FD                      STY   PAG3
0348:                ;
0348:B1 FB           LOOP2      LDA   (PAG2),Y       Get pic byte
034A:48                         PHA                  and save on stack
034B:A5 FF                      LDA   STEP           Get current STEP value
034D:D1 FD                      CMP   (PAG3),Y       STEP >= Random mask?
034F:B0 04     0355             BCS   NOMASK         Yes, use pic byte
0351:68                         PLA                  No, discard pic byte, and
0352:                ;
0352:          0000  TEST       IFNE  DOT            ;Do DOT procedure?
 S                              LDA   #0             Yes, use a clear byte
0352:                           ELSE  ;              Do MASK procedure
0352:B1 FD                      LDA   (PAG3),Y       Get random byte
0354:                           FIN
0354:48                         PHA                  and save on stack
0355:                ;
0355:68              NOMASK     PLA                  Recover byte
0356:91 F9                      STA   (PAG1),Y       and store to display
0358:C8                         INY                  Done this line?
0359:D0 ED     0348             BNE   LOOP2          No, go again
035B:                ;
035B:E6 FE           NXTBYT1    INC   PAG3+1         Next line
035D:E6 FC                      INC   PAG2+1
035F:E6 FA                      INC   PAG1+1
0361:A5 FA                      LDA   PAG1+1
0363:C9 40                      CMP   #HPAG2         Done?
0365:90 E1     0348             BCC   LOOP2          No, go again
0367:60                         RTS                  Yes, return to caller
0368:                ;
0368:                *================================
0368:                ;                INVERT
0368:                *================================
0368:                ;
0368:                ; CALL 872 from BASIC.
0368:                ;
0368:A9 40           INVERT     LDA   #HPAG2         pic is on page 2
036A:85 FC                      STA   PAG2+1
036C:A0 00                      LDY   #0
036E:84 FB                      STY   PAG2
0370:                ;
0370:B1 FB           LOOP3      LDA   (PAG2),Y       get byte
0372:49 7F                      EOR   #%01111111     complement it
0374:91 FB                      STA   (PAG2),Y       and put back
0376:C8                         INY                  More on this line?
0377:D0 F7     0370             BNE   LOOP3          Yes, go
0379:E6 FC                      INC   PAG2+1         No, next line
```

Listing 2, continued

```
037B:A5 FC                     LDA   PAG2+1
037D:C9 60                     CMP   #HPAG3        Done?
037F:90 EF     0370            BCC   LOOP3         Yes, do it
0381:60                        RTS                 Else, return to caller
0382:                ;
0382:                *=============================
0382:                ;           REFLECT
0382:                *=============================
0382:                ;
0382:                ; With the picture to be reflected on
0382:                ; HIRES page 2, from BASIC POKE 230, 64
0382:                ; and then CALL 898.
0382:                ;
0382:          0026  SCREEN    EQU   $26           HIRES pointer
0382:          F411  HPOSN     EQU   $F411         position calculator
0382:                ;
0382:A9 00           REFLECT   LDA   #0            use PAG3 as a counter
0384:85 FD                     STA   PAG3
0386:                ;
0386:A2 00           REFLOOP   LDX   #0            point to left edge
0388:A0 00                     LDY   #0
038A:A5 FD                     LDA   PAG3          get vertical line
038C:20 11 F4                  JSR   HPOSN         calculate byte
038F:98                        TYA                 calculate right edge
0390:18                        CLC
0391:69 27                     ADC   #39
0393:85 F9                     STA   PAG1          PAG1 as right edge index
0395:                ;
0395:A9 00           LOOPIT    LDA   #0            clear STEP (used as
0397:85 FF                     STA   STEP          a temporary buffer)
0399:B1 26                     LDA   (SCREEN),Y    get left edge byte
039B:A2 06                     LDX   #6            rotate it
039D:6A              LOOP4     ROR   A
039E:26 FF                     ROL   STEP
03A0:CA                        DEX
03A1:10 FA     039D            BPL   LOOP4
03A3:98                        TYA                 save left index
03A4:48                        PHA
03A5:                ;
03A5:A4 F9           DORIGHT   LDY   PAG1          get right index
03A7:B1 26                     LDA   (SCREEN),Y    get right edge byte
03A9:48                        PHA                 save on stack
03AA:A5 FF                     LDA   STEP          get left rotated byte
03AC:91 26                     STA   (SCREEN),Y    and swap left -> right
03AE:A9 00                     LDA   #0            clear STEP (buffer)
03B0:85 FF                     STA   STEP
03B2:68                        PLA                 recover right edge byte
03B3:A2 06                     LDX   #6            rotate it
03B5:6A              LOOP5     ROR   A
03B6:26 FF                     ROL   STEP
03B8:CA                        DEX
03B9:10 FA     03B5            BPL   LOOP5
03BB:68                        PLA                 recover left index
03BC:A8                        TAY
03BD:A5 FF                     LDA   STEP          get right rotated byte
03BF:91 26                     STA   (SCREEN),Y    swap right -> left
03C1:                ;
03C1:C8              NXTPAIR   INY                 next pair of pic bytes
03C2:C6 F9                     DEC   PAG1
03C4:C4 F9                     CPY   PAG1          done this line?
03C6:90 CD     0395            BCC   LOOPIT        No, go again
03C8:                ;
03C8:E6 FD           NXT40     INC   PAG3          Else, next line
03CA:A5 FD                     LDA   PAG3          get vertical line
03CC:C9 C0                     CMP   #192          Done?
03CE:90 B6     0386            BCC   REFLOOP       No, go again
03D0:60                        RTS                 Else, return to caller
```

larly, stimuli may be overlaid on the same or different background scenes to investigate, for example, the effects of context on perception.

Other Dependent Variables

In addition to number of keypresses (or, equivalently, percentage clarified) to correct identification, the clarification task lends itself naturally to a number of other dependent measures. As mentioned, time to correct identification also may be recorded. Combining these measures produces a third dependent variable, time per keypress, that is in logic independent of the original two. Mean times per keypress (in units of a counting loop) for the mask-clarification experiment presented earlier were 5.6, 7, and 8.4 for *identical*, *same name*, and *different* picture types, respectively. A significant effect of picture-type was evident [$F(2,8) = 6.96$, $MSe = 1.35$, $p < .0177$]. Subjects studied *identical* pictures for less time on each trial before advancing to the next trial than they did for either *same name* or *different* pictures, and spent the most time per trial studying *different* pictures. Thus, prior exposure to a particular picture not only increases the amount of noise subjects can tolerate for correct identification (as shown by the measure of percen-

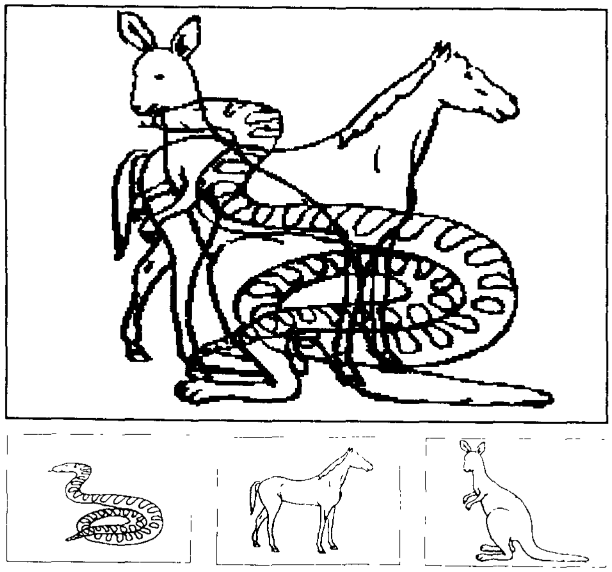

Figure 3. An example of overlapping pictures, producing stimuli similar to those used by Ghent (1956).

tage clarified for correct identification), but also reduces the amount of study time required to identify the picture through the noise.

Psychophysical functions may be obtained from the clarification task by modifying the task so that different sets of pictures are shown at different fixed levels of clarification (e.g., 10%, 20%, etc.). The identification accuracy (number or percentage of pictures correctly identified) at each level of clarification is then recorded. From these data, the common psychophysical identification and recognition thresholds may be computed. (See Uttal, 1975, for examples of this approach using a related procedure.)

Perceptual Identification of Letters and Words

Although for most of our research with the clarification routines, we have used pictures, the same routines may be applied to letter and word stimuli displayed on the Apple's high-resolution screen. In this way, for example, degraded letter stimuli similar to those used in Warrington and James's (1967) incomplete letters test (similar to dot clarification) and in Warrington and Taylor's (1973) figure-ground test (similar to mask clarification), and degraded word stimuli similar to those developed by Barber and de la Mahotière (1982) and by Johnston, Dark, and Jacoby (a version of mask clarification; 1985) may easily be created. To effect these stimuli, an Applesoft BASIC shape table containing a replica of the complete standard character set on the Apple DMP (or, equivalently, the Apple ImageWriter) printer was developed. These letters are then drawn on the high-resolution screen, where they may be subjected to the same procedures, including mask and dot clarification, as any other high-resolution display. In fact, because a shape table is used, the scaling and rotation features inherent in Applesoft BASIC may be applied to the letters and, when coupled with the reflection utility, may be used to produce rotated and reflected letter and word stimuli similar to those of Kolers (1976).

The Clarification Package

The complete package of clarification routines and associated support software operating under Apple's ProDOS environment is available from the authors. Included in the package is the program KRUNCHIT which is a stand-alone menu-driven program used to compress high-resolution pictures created on the Apple II. Its features include an extensive HELP function and facilities to edit and modify high-resolution pictures before compressing them. Also included in the package are the source and object files of each of the routines discussed in this article, a subdirectory of programs providing examples of different experimental procedures, the DMP character-set shape table, and a diskette containing 60 compressed line drawings. The package may be obtained at no charge by sending two Apple II compatible floppy diskettes to John R. Vokey, Department of Psychology, University of Lethbridge, Lethbridge, Alberta, Canada T1K 3M4.

REFERENCES

Barber, P., & de la Mahotière, C. (1982). Ease of identifying words degraded by visual noise. *British Journal of Psychology*. **73**, 371-381

Brooks, L. R., Jacoby, L. L., & Whittlesea, B. W. A. (in preparation). The influence of specific familiarity on picture identification.

Bruner, J. S., & Potter, M. C. (1964) Interference in visual recognition. *Science*, **144**, 424-425.

Dunn, L. M. (1965). *Peabody picture vocabulary test* Circle Pines, MN: American Guidance Service, Inc.

Ghent, L. (1956). Perception of overlapping and embedded figures by children of different ages. *American Journal of Psychology*, **69**, 575-587.

Gollin, E. S. (1960). Developmental studies of visual recognition of incomplete objects. *Perceptual & Motor Skills*, **11**, 289-298.

Green, D. M., & Swets, J. A. (1966). *Signal detection theory and psychophysics*. New York: Wiley.

Kaner, H. C., & Vokey, J. R. (1984). A better random number generator. *Micro*, **72**, 26-35.

Jacoby, L. L., & Brooks, L. R. (1984). Non-analytic cognition: Memory, perception, and concept learning. In G. H. Bower (Ed.), *The psychology of learning and motivation: Advances in research and theory* (Vol. 18). New York: Academic Press.

Johnston, W. A., Dark, V. J., & Jacoby, L. L. (1985) Perceptual fluency and recognition judgments. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **11**, 3-11.

Kolers, P. A. (1976) Reading a year later. *Journal of Experimental Psychology: Human Learning & Memory*, **2**, 554-565

Mooney, C. M. (1956). Closure with negative afterimages under flickering light. *Canadian Journal of Psychology*, **10**, 191-199.

Mooney, C. M. (1957). Age in the development of closure ability in children. *Canadian Journal of Psychology*, **2**, 219-226.

Snodgrass, J. G., & Vanderwart, M. (1980). A standardized set of 260 pictures. Norms for name agreement, image agreement, familiarity, and visual complexity. *Journal of Experimental Psychology: Human Learning and Memory*, **6**, 174-215.

Sparks, D. (1983) RND is fatally flawed *Call A.P.P.L.E.*, **6**, 29-34.

SPERLING, G (1960). The information available in brief visual presentations. *Psychological Monographs*, **7**(11).

UTTAL, W. R (1975). *An autocorrelation theory of form detection* Hillsdale, NJ: Erlbaum

WARREN, C , & MORTON, J. (1982). The effects of priming on picture recognition. *British Journal of Psychology*, **73**, 117-129.

WARRINGTON, E K. (1982). Neuropsychological studies of object identification. *Philosophical Transactions of the Royal Society of London*, **298**, 15-33.

WARRINGTON, E. K , & JAMES, M. (1967). Disorders of visual perception in patients with localized cerebral lesions. *Neuropsychologia*, **5**, 253-266.

WARRINGTON, E. K., & TAYLOR, A. M. (1973). The contribution of the right parietal lobe to object recognition. *Cortex*, **9**, 152-164.

NOTE

1 Applesoft BASIC's RND function is flawed; the lower 8 bits of both the multiplier and the additive constant of the generator are missing, resulting in the generator's falling into short repetitive cycles rather than completing its theoretical period of a trillion-plus numbers before repeating (see, e g., Sparks, 1983). Kaner and Vokey (1984) provide three independently addressable random-number generators for the Apple II, interfaced to Applesoft BASIC via the USR function, that may be used to correct the problem.

(Manuscript received June 5, 1985;
revision accepted for publication December 4, 1985.)