MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

Working Paper No. 287 April 14, 1986

GROK Doc
An Image Display Tool

Jim Little

Abstract.
The image display tool GROK provides a facility for displaying images on the black-and-white screen of a Symbolics 3600 monitor. It allows display of images and their manipulation through a special window it manages. Images become objects in that window, and are handled by a variety of routines accessible by mouse selection from window menus. GROK is an outgrowth of two programs- Keith Nishihara's GREY*, which provided the concept of an image manipulation and display program for black-and-white screens, and Margaret Fleck's GREYCROK, which formed the nucleus from which GROK mutated. Many of the functions in GROK are lifted directly from GREYCROK.

1 Introduction

The only purpose of this paper is to provide enough documentation for GROK so that you can get it going on a Lisp Machine and use it. GROK resides in file *pig:[vision.utils]GROK.* GROK lives in package VISION, which has the nicknames VIS and V. The word *grok* originates in Robert Heinlein's "Stranger in a Strange Land" [1961]:

> "You grok," Smith repeated firmly. "I am explain. I did not have the word. You grok. Anne groks. I grok. The grasses under my feet grok in happy beauty."

grok is a flexible word, and GROK is intended to be a flexible image display tool. *grok* has connotations of understanding, empathy, and insight, just as GROK is designed to lead us to undertanding and insight. The word *grok* is best understood in context, through use, as is GROK.

1.1 GROK Mutating

GROK is always under development, under the pressure of skilled users who want only the best for their work. Therefore, this manual will almost certainly be immediately out of date. Every effort will be made to keep GROK downward compatible. A log of changes to GROK with some description of their function will be kept in file *pig:[vision.utils]GROKINFO.txt.*

1.2 A Note about Notation

This paper uses a format like that of the Lisp Machine Manual (e.g. Symbolics [1985]): the documentation for each function, macro, variable, or method starts with a header line containing its name, arguments, and type. For clarity, the default values of optional arguments are often omitted from the header line and are described in the documentation instead. You should *not* assume that the absence of a default value of an optional variable in the header line means that the default value is **nil**.

GROK is the general name of the utility. **grok** is the name of the Lisp function which constructs the objects depicting images on the Lisp Machine screen.

2 A Sample of GROK Use

Here's a small session of using GROK, from loading it, to using it. The documentation for the functions will follow later, so just read along and save the questions for later.

```
(load "pig:<vision.utils>grok")
```

This loads GROK.

```
(v:make-image-display-frame 0.7 t t)
```

This makes an *image-display-window* as part of a frame, occupying the bottom 70% of the frame. Now, read in or generate an image. Here we use the image of a pear. **pear** is bound to an *array* containing the image.

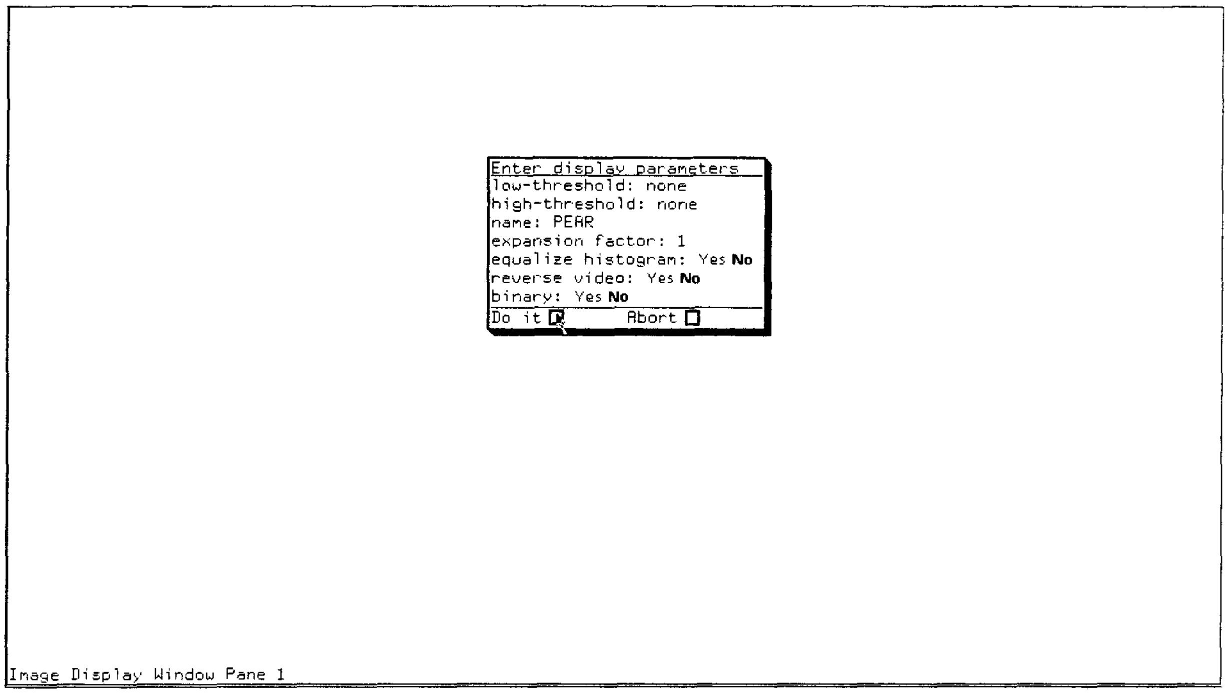
```
(v:show pear)
```

Figure 1 shows a standard window system *choose-variable-values* menu popped up in the Image Display Window Pane for setting the display parameters. Figure 2 depicts an *image-object* displaying the pear.

```
(v:show peare)
```

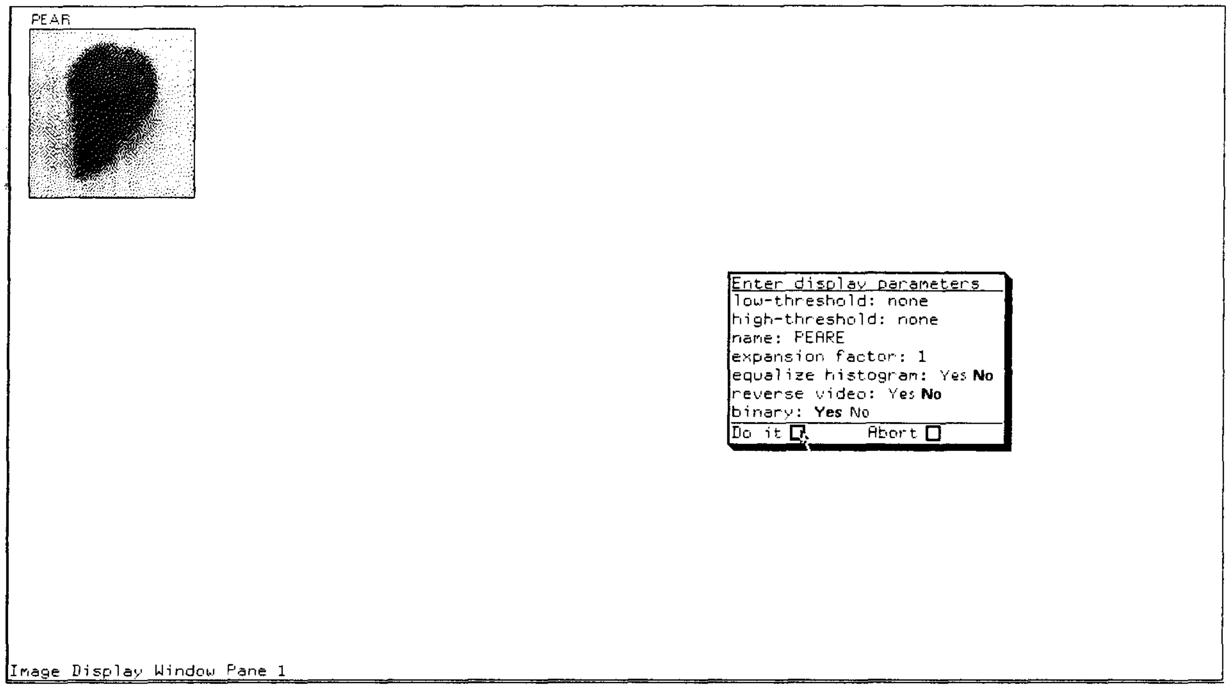
This displays the edge image of the pear in an *image-object.* GROK recognizes that **peare** is an *art-1b* array, and sets the **binary** switch to **t**, as shown in Figure 2. The "Do It" selection is about to be activated. In Figure 3, the pear edge image, labeled PEARE, is shown. The image is shown with 1's as black, since ***g-reverse-binary-switch*** defaults to **t**.

Next to that *image-object* is the *"Image display operations"* menu (which can be activated with the right mouse button), from which "Extract" is being selected. In Figure 4, a box indicating the image portion to be extracted appears on the *image-object* PEAR; its position is controlled by positioning its upper left corner with the mouse, clicking, and then rubber-boxing its lower-right hand corner with the mouse. The box can then be moved around within the image. When a click is read, a menu pops up for entering a name for the *image-object* (see Figure 5) with a default name, and a name can be entered. It is finished by selecting "Do it". Note that GROK automagically finds a new position for the *image-object*, and draws a



Enter display parameters
low-threshold: none
high-threshold: none
name: PEAR
expansion factor: 1
equalize histogram: Yes [✓ No]
reverse video: Yes [✓ No]
binary: Yes [✓ No]

Figure 1: Image Display Parameters Menu



Enter display parameters
low-threshold: none
high-threshold: none
name: PEARE
expansion factor: 1
equalize histogram: Yes [✓ No]
reverse video: Yes [✓ No]
binary: [✓ Yes] No

Figure 2: Pear Display, with Menu

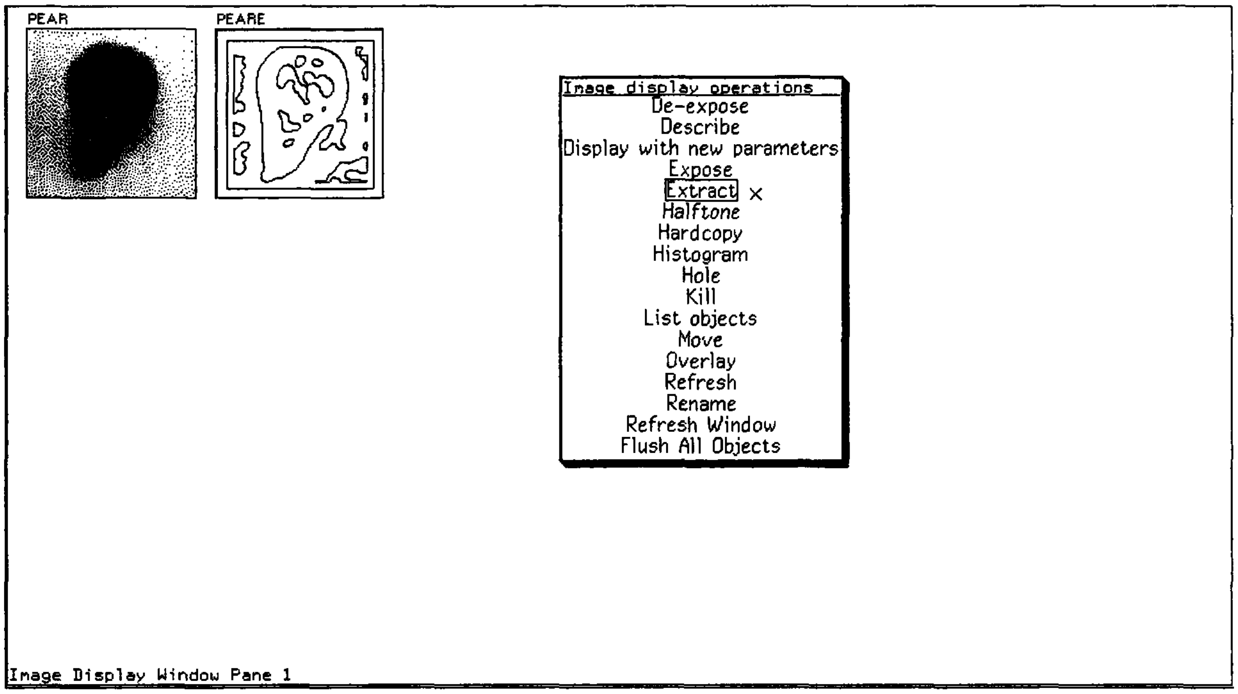


Figure 3: Selecting an Operation

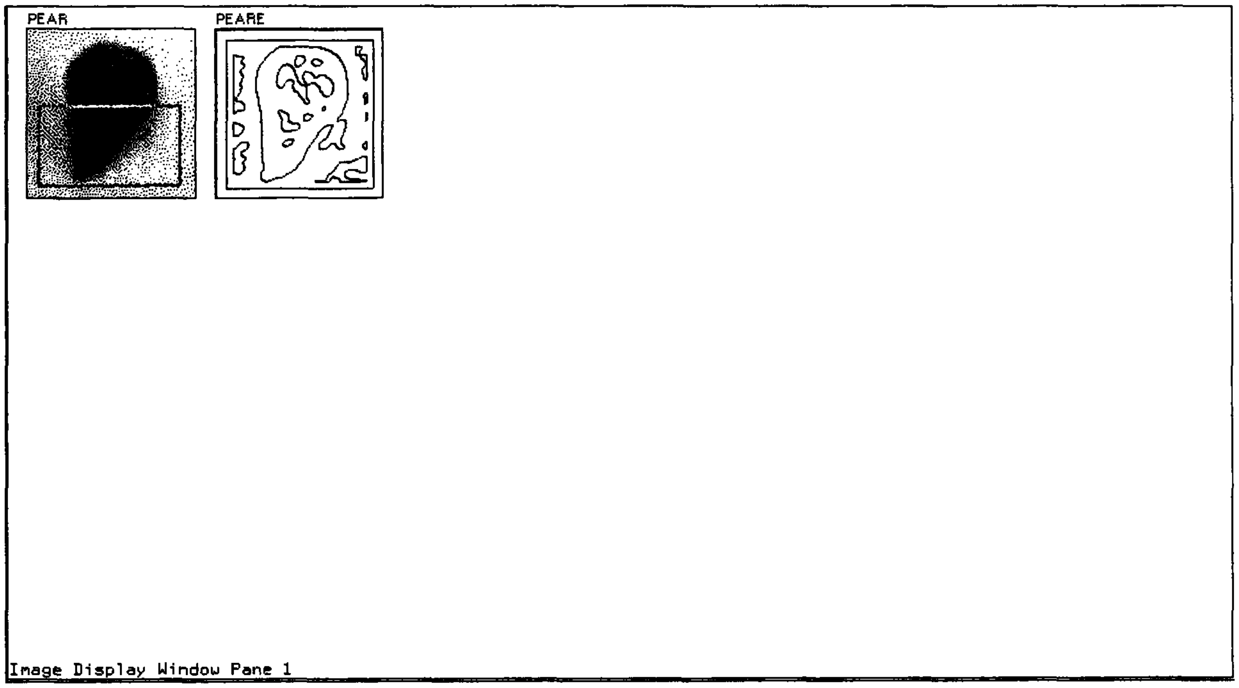


Figure 4: Extracting an Image Portion

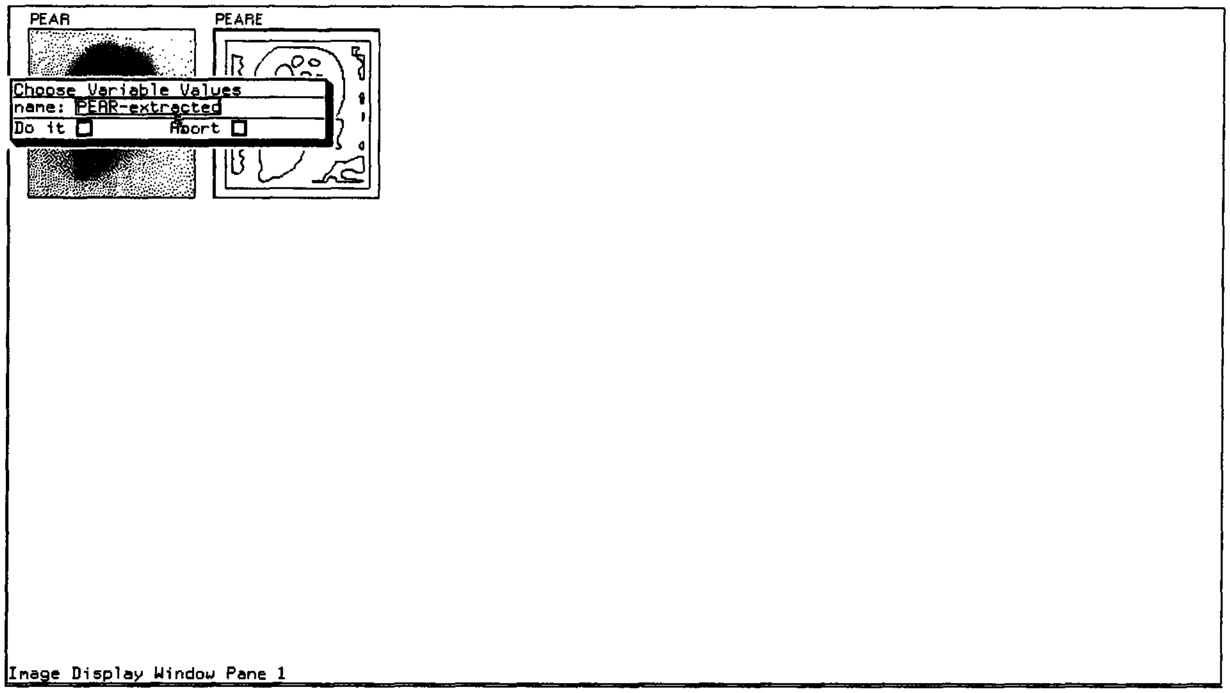


Figure 5: Entering an Image-Object Name

box on the screen at that position. You can then move the box to a more suitable position with the mouse, or click to have the object drawn at the current position.

Overlay generates an image by combining the bit-map of two *image-objects* using a *bitblt* operation with a selected ALU. The Image display operations menu is invoked by right-clicking on one *image-object* (A). "Overlay" can be selected from the operations menu, and then clicking on the second *image-object* (B), as directed by the info on the mouse-line (at the bottom of the window), activates either the left or middle ALU, or displays the overlay-ALU menu (right click). Figure 6 shows the overlay-ALU menu. Any of the ALU's in the menu can be selected. The resulting *image-object* is depicted in Figure 7.

To the right of *image-object* PEARE-over-PEAR is a description of PEAR generated by selecting the menu of image operations with a right click on PEAR, and selecting "Describe".

Figure 8 shows the "Pixels" operation, selected by clicking left-twice on an *image-object*. A box is drawn around the location of the mouse, on the *image-object*, and a menu pops up to allow direction of the output either to a temporary window, later described as a *scratch-window*, or to a permanent *image-object*. Figure 9 shows such an object, with the array of pixel values printed in it.

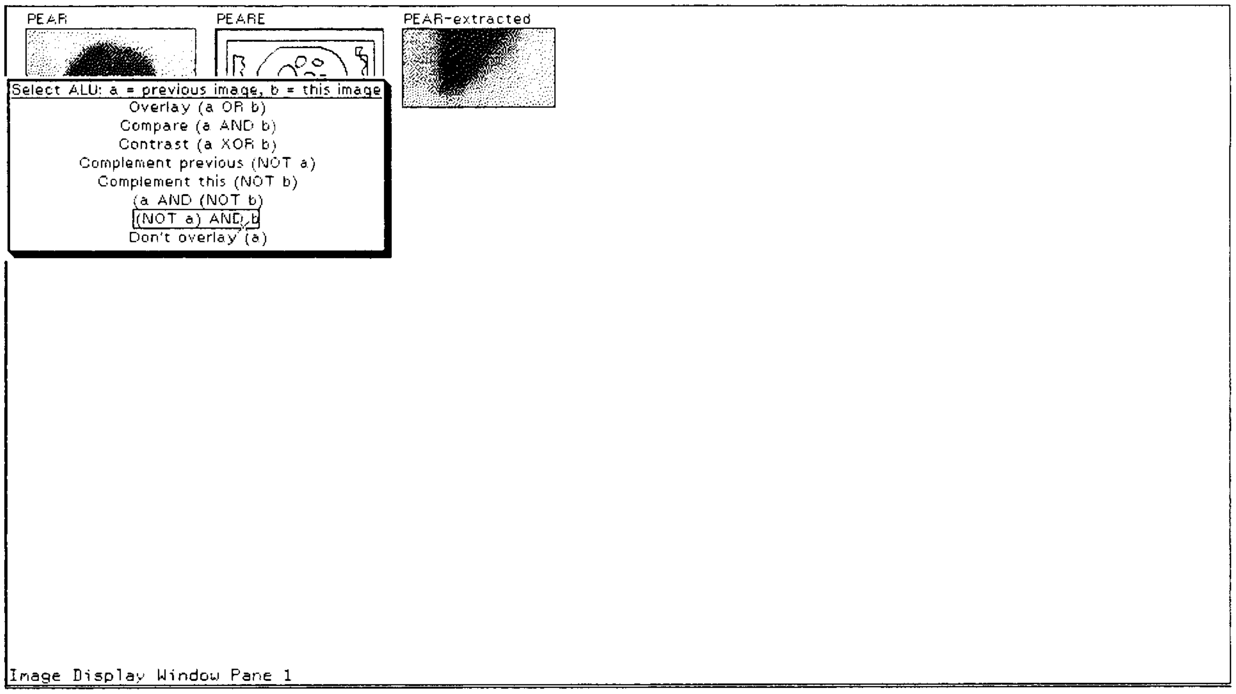
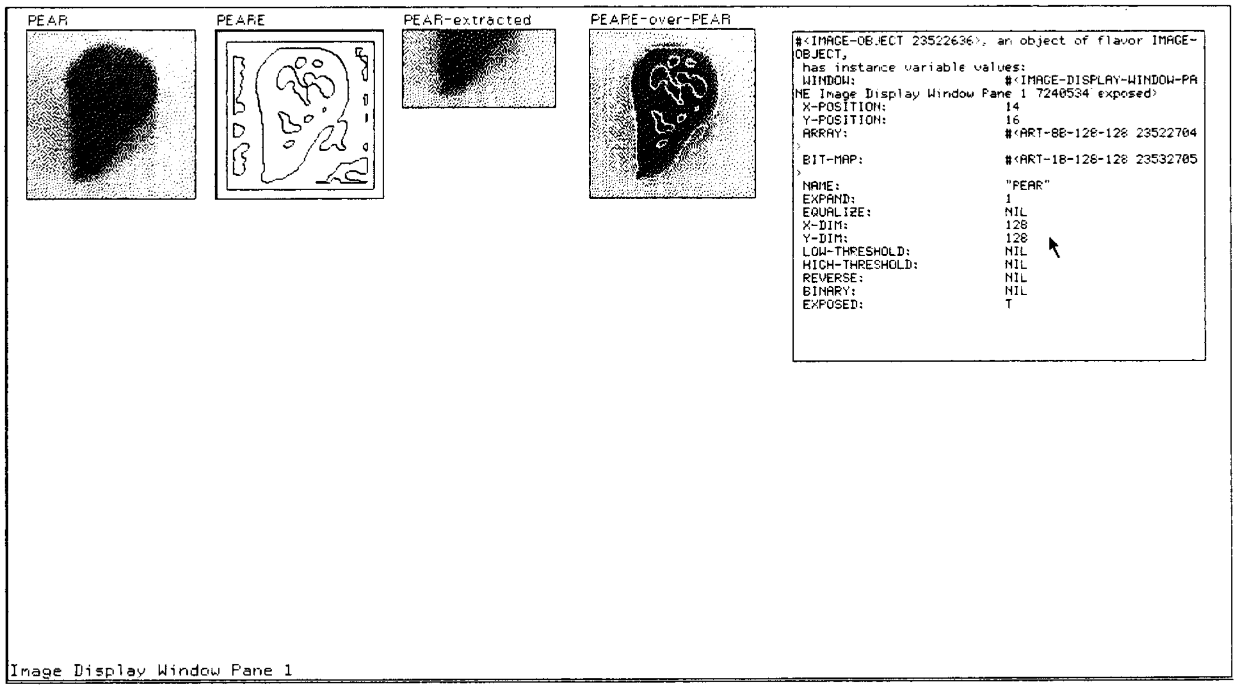
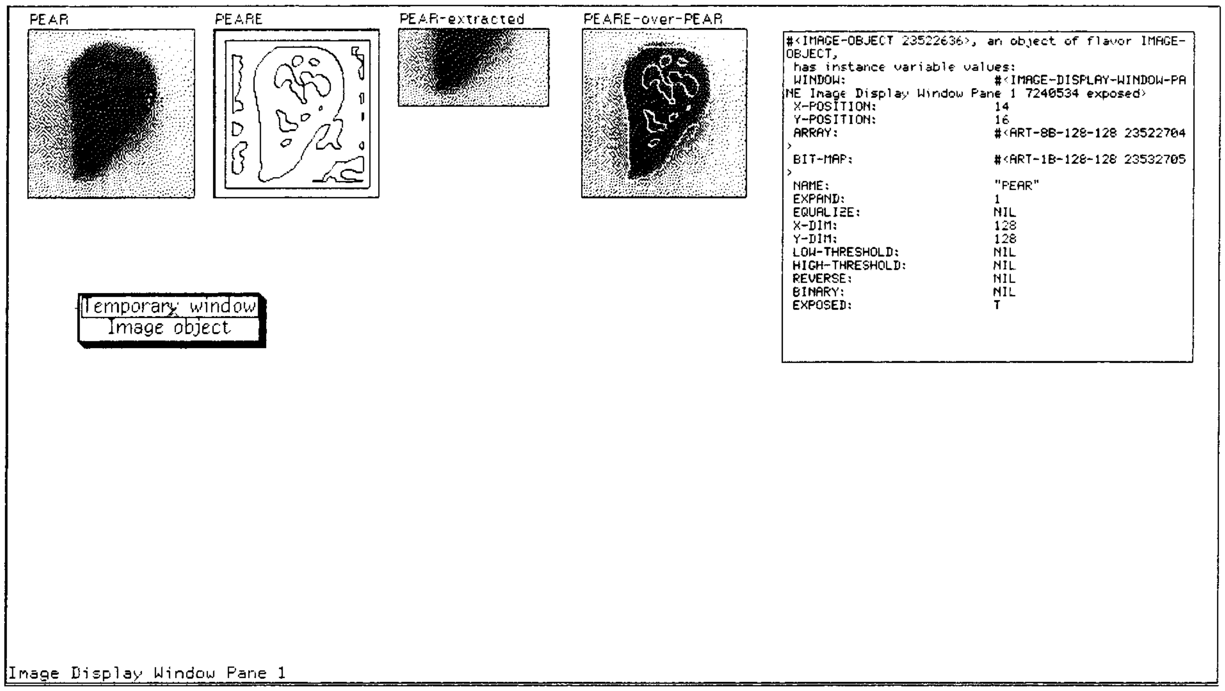


Figure 6: Selecting an Overlay ALU



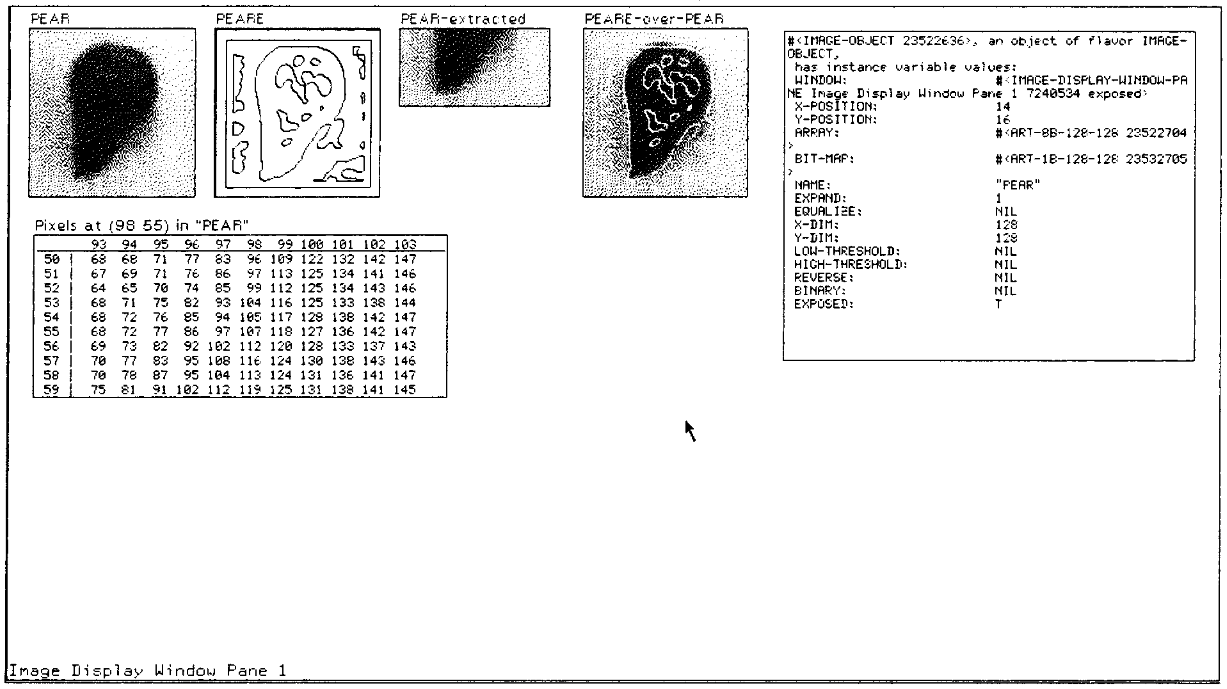
```
#<IMAGE-OBJECT 23522636>, an object of flavor IMAGE-
OBJECT,
 has instance variable values:
 WINDOW:                       #<IMAGE-DISPLAY-WINDOW-PA
NE Image Display Window Pane 1 7240534 exposed>
 X-POSITION:                   14
 Y-POSITION:                   16
 ARRAY:                        #<ART-8B-128-128 23522704
>
 BIT-MAP:                      #<ART-1B-128-128 23532705
>
 NAME:                         "PEAR"
 EXPAND:                       1
 EQUALIZE:                     NIL
 X-DIM:                        128
 Y-DIM:                        128
 LOW-THRESHOLD:                NIL
 HIGH-THRESHOLD:               NIL
 REVERSE:                      NIL
 BINARY:                       NIL
 EXPOSED:                      T
```

Figure 7: An Overlay and a Description



```
#<IMAGE-OBJECT 23522636>, an object of flavor IMAGE-
OBJECT,
 has instance variable values:
 WINDOW:                     #<IMAGE-DISPLAY-WINDOW-PA
NE Image Display Window Pane 1 7240534 exposed>
 X-POSITION:                 14
 Y-POSITION:                 16
 ARRAY:                      #<ART-8B-128-128 23522704
>
 BIT-MAP:                    #<ART-1B-128-128 23532705
>
 NAME:                       "PEAR"
 EXPAND:                     1
 EQUALIZE:                   NIL
 X-DIM:                      128
 Y-DIM:                      128
 LOW-THRESHOLD:              NIL
 HIGH-THRESHOLD:             NIL
 REVERSE:                    NIL
 BINARY:                     NIL
 EXPOSED:                    T
```

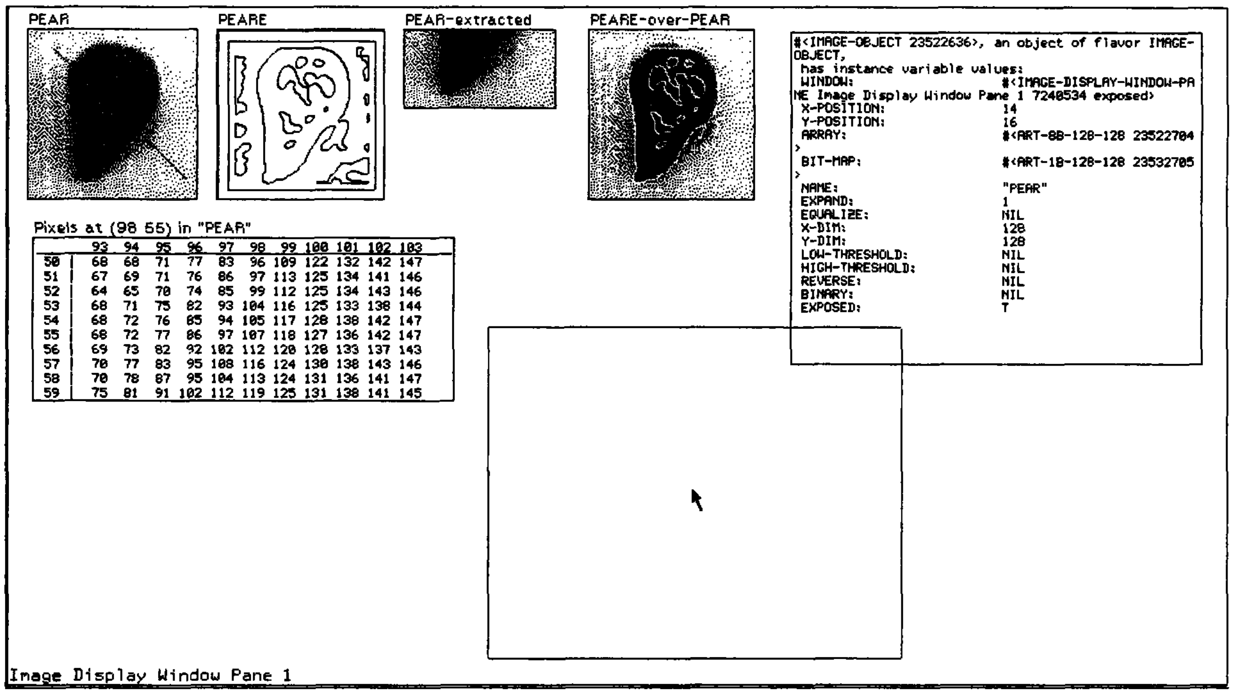
Figure 8: Displaying Nearby Pixels



```
Pixels at (98 55) in "PEAR"
         93  94  95  96  97  98  99 100 101 102 103
 50 |    68  68  71  77  83  96 109 122 132 142 147
 51 |    67  69  71  76  86  97 113 125 134 141 146
 52 |    64  65  70  74  85  99 112 125 134 143 146
 53 |    68  71  75  82  93 104 116 125 133 138 144
 54 |    68  72  76  85  94 105 117 128 138 142 147
 55 |    68  72  77  86  97 107 118 127 136 142 147
 56 |    69  73  82  92 102 112 120 128 133 137 143
 57 |    70  77  83  95 108 116 124 130 138 143 146
 58 |    70  78  87  95 104 113 124 131 136 141 147
 59 |    75  81  91 102 112 119 125 131 138 141 145
```

```
#<IMAGE-OBJECT 23522636>, an object of flavor IMAGE-
OBJECT,
 has instance variable values:
 WINDOW:                     #<IMAGE-DISPLAY-WINDOW-PA
NE Image Display Window Pane 1 7240534 exposed>
 X-POSITION:                 14
 Y-POSITION:                 16
 ARRAY:                      #<ART-8B-128-128 23522704
>
 BIT-MAP:                    #<ART-1B-128-128 23532705
>
 NAME:                       "PEAR"
 EXPAND:                     1
 EQUALIZE:                   NIL
 X-DIM:                      128
 Y-DIM:                      128
 LOW-THRESHOLD:              NIL
 HIGH-THRESHOLD:             NIL
 REVERSE:                    NIL
 BINARY:                     NIL
 EXPOSED:                    T
```

Figure 9: Nearby Pixels Values



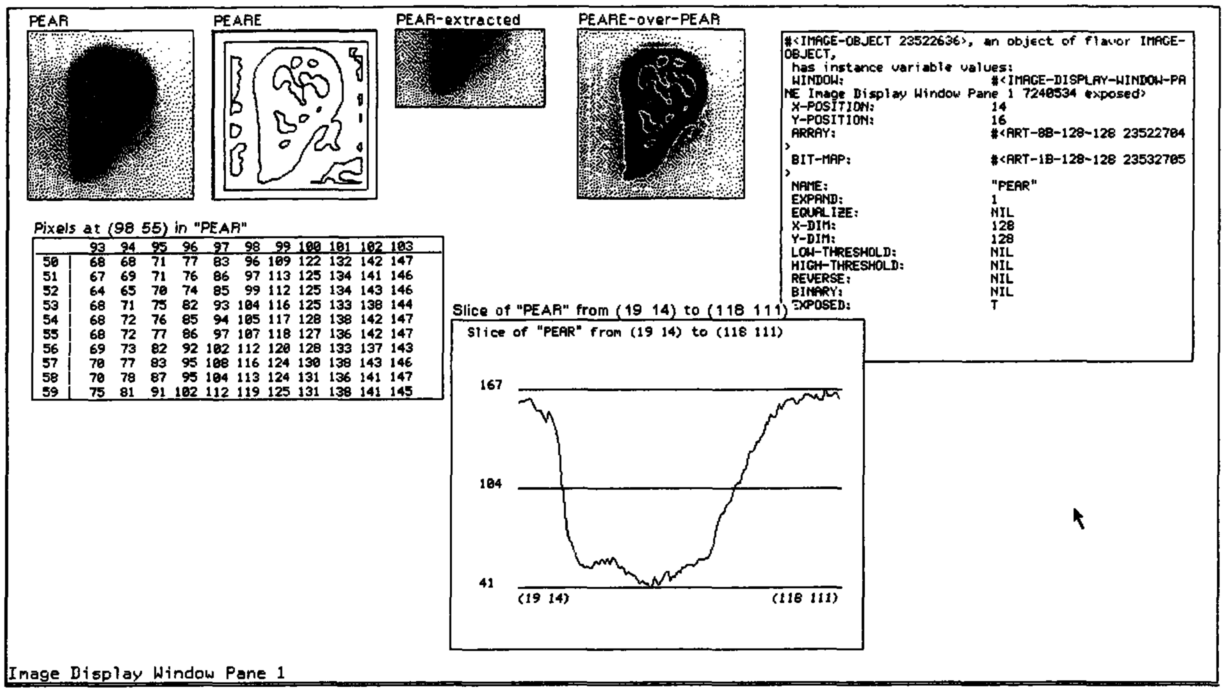
```
Pixels at (98 55) in "PEAR"
        93  94  95  96  97  98  99 100 101 102 103
  50 |  68  68  71  77  83  96 109 122 132 142 147
  51 |  67  69  71  76  86  97 113 125 134 141 146
  52 |  64  65  70  74  85  99 112 125 134 143 146
  53 |  68  71  75  82  93 104 116 125 133 138 144
  54 |  68  72  76  85  94 105 117 128 138 142 147
  55 |  68  72  77  86  97 107 118 127 136 142 147
  56 |  69  73  82  92 102 112 120 128 133 137 143
  57 |  70  77  83  95 108 116 124 130 138 143 146
  58 |  70  78  87  95 104 113 124 131 136 141 147
  59 |  75  81  91 102 112 119 125 131 138 141 145
```

```
#<IMAGE-OBJECT 23522636>, an object of flavor IMAGE-
OBJECT,
 has instance variable values:
 WINDOW:                   #<IMAGE-DISPLAY-WINDOW-PA
NE Image Display Window Pane 1 7240534 exposed>
 X-POSITION:               14
 Y-POSITION:               16
 ARRAY:                    #<ART-8B-128-128 23522704
>
 BIT-MAP:                  #<ART-1B-128-128 23532705
>
 NAME:                     "PEAR"
 EXPAND:                   1
 EQUALIZE:                 NIL
 X-DIM:                    128
 Y-DIM:                    128
 LOW-THRESHOLD:            NIL
 HIGH-THRESHOLD:           NIL
 REVERSE:                  NIL
 BINARY:                   NIL
 EXPOSED:                  T
```

Figure 10: Slicing an Image

Next, a “Slice image” operation is in progress (Figure 10 bottom), selected by clicking middle-once, with a diagonal slice through PEAR at the selected position, and a box showing the position of the output object. Figure 11 shows the plot of the slice through the PEAR *image-object* at that position. Finally, the bottom of Figure 12 shows a histogram of PEAR, activated by selecting “Histogram” from the *“Image display operations”* menu.

A hole through one image to another can be selected from the *“Image display operations”* menu. If the mouse is not on an *image-object*, a menu with the names of the objects in the window will pop up, and the appropriate object can be selected (Figure 13). Next, a box can be moused on the object as in the “Extract” operation. When the box is finished, a menu will pop up to select an image to be seen through the hole. The result of seeing PEARE through PEAR is in Figure 14.

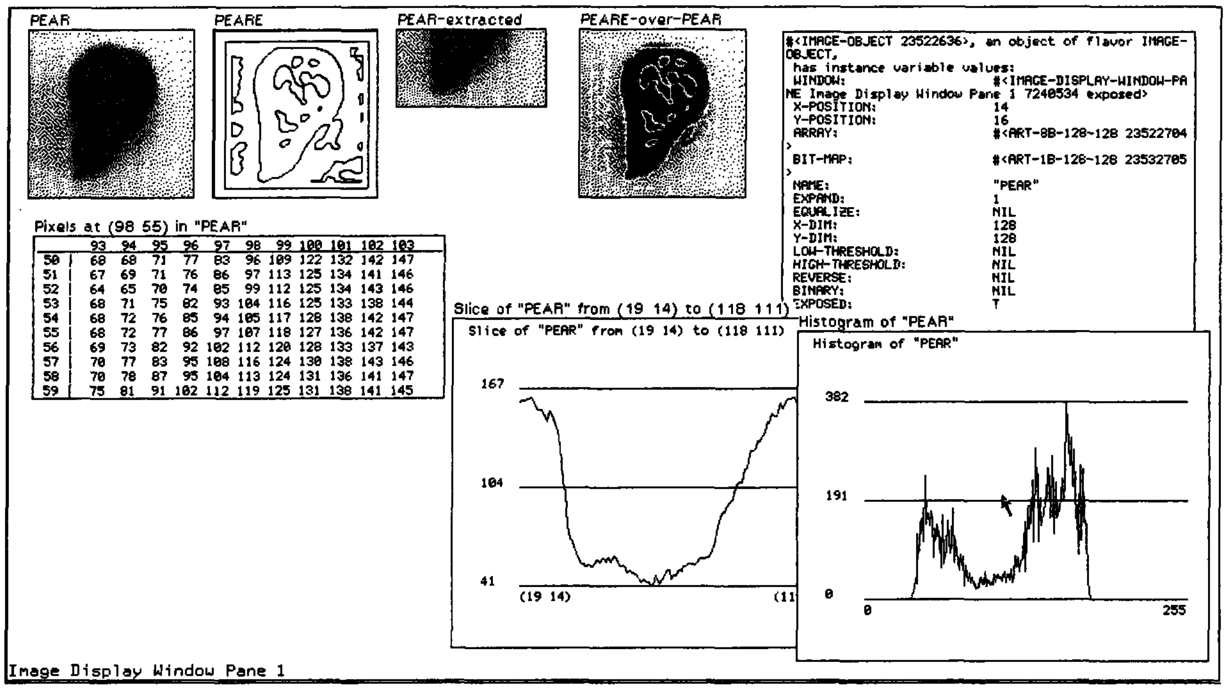

PEAR
PEARE
PEAR-extracted
PEARE-over-PEAR
Slice of "PEAR" from (19 14) to (118 111)
Slice of "PEAR" from (19 14) to (118 111)
167
104
41
(19 14)
(118 111)
Image Display Window Pane 1

Pixels at (98 55) in "PEAR"

	93	94	95	96	97	98	99	100	101	102	103
50	68	68	71	77	83	96	109	122	132	142	147
51	67	69	71	76	86	97	113	125	134	141	146
52	64	65	70	74	85	99	112	125	134	143	146
53	68	71	75	82	93	104	116	125	133	138	144
54	68	72	76	85	94	105	117	128	138	142	147
55	68	72	77	86	97	107	118	127	136	142	147
56	69	73	82	92	102	112	120	128	133	137	143
57	70	77	83	95	108	116	124	130	138	143	146
58	70	78	87	95	104	113	124	131	136	141	147
59	75	81	91	102	112	119	125	131	138	141	145

```
#<IMAGE-OBJECT 23522636>, an object of flavor IMAGE-
OBJECT,
 has instance variable values:
 WINDOW:                    #<IMAGE-DISPLAY-WINDOW-PA
NE Image Display Window Pane 1 7240534 exposed>
 X-POSITION:                14
 Y-POSITION:                16
 ARRAY:                     #<ART-8B-128~128 23522704
>
 BIT-MAP:                   #<ART-1B-128~128 23532705
>
 NAME:                      "PEAR"
 EXPAND:                    1
 EQUALIZE:                  NIL
 X-DIM:                     128
 Y-DIM:                     128
 LOW-THRESHOLD:             NIL
 HIGH-THRESHOLD:            NIL
 REVERSE:                   NIL
 BINARY:                    NIL
 EXPOSED:                   T
```

Figure 11: Slicing an Image

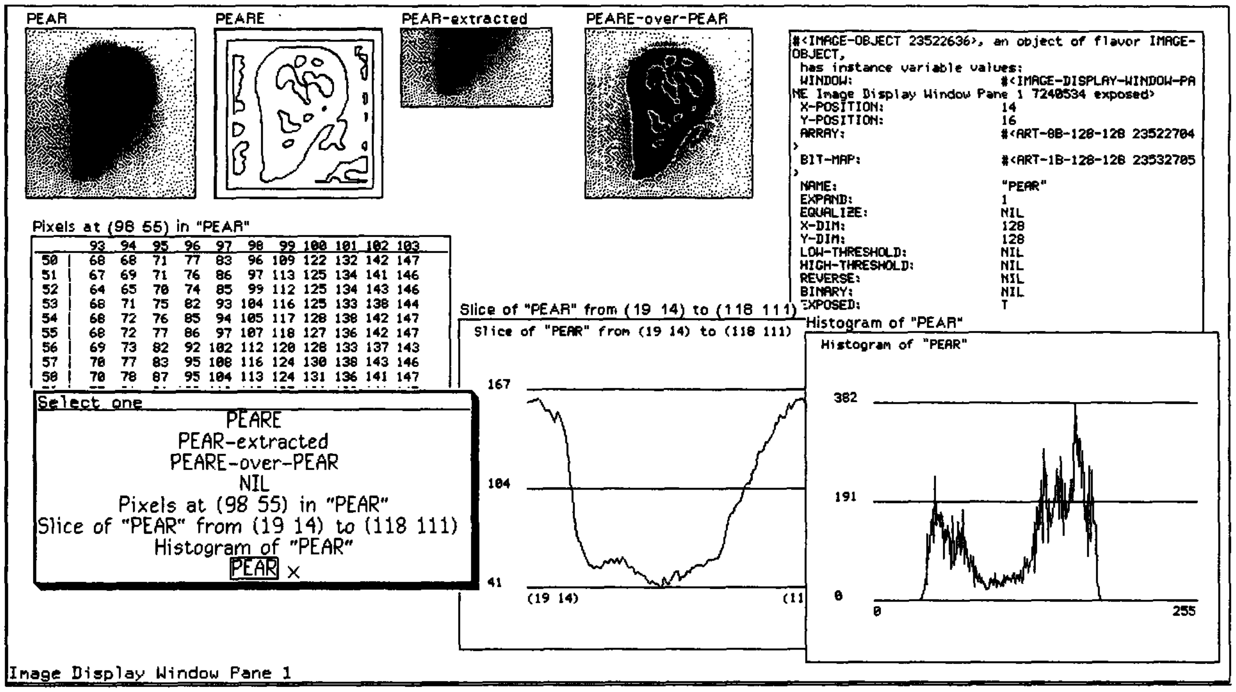

PEAR
PEARE
PEAR-extracted
PEARE-over-PEAR
Slice of "PEAR" from (19 14) to (118 111)
Slice of "PEAR" from (19 14) to (118 111)
167
104
41
(19 14)
Histogram of "PEAR"
Histogram of "PEAR"
382
191
0
0
255
Image Display Window Pane 1

Pixels at (98 55) in "PEAR"

	93	94	95	96	97	98	99	100	101	102	103
50	68	68	71	77	83	96	109	122	132	142	147
51	67	69	71	76	86	97	113	125	134	141	146
52	64	65	70	74	85	99	112	125	134	143	146
53	68	71	75	82	93	104	116	125	133	138	144
54	68	72	76	85	94	105	117	128	138	142	147
55	68	72	77	86	97	107	118	127	136	142	147
56	69	73	82	92	102	112	120	128	133	137	143
57	70	77	83	95	108	116	124	130	138	143	146
58	70	78	87	95	104	113	124	131	136	141	147
59	75	81	91	102	112	119	125	131	138	141	145

```
#<IMAGE-OBJECT 23522636>, an object of flavor IMAGE-
OBJECT,
 has instance variable values:
 WINDOW:                    #<IMAGE-DISPLAY-WINDOW-PA
NE Image Display Window Pane 1 7240534 exposed>
 X-POSITION:                14
 Y-POSITION:                16
 ARRAY:                     #<ART-8B-128~128 23522704
>
 BIT-MAP:                   #<ART-1B-128~128 23532705
>
 NAME:                      "PEAR"
 EXPAND:                    1
 EQUALIZE:                  NIL
 X-DIM:                     128
 Y-DIM:                     128
 LOW-THRESHOLD:             NIL
 HIGH-THRESHOLD:            NIL
 REVERSE:                   NIL
 BINARY:                    NIL
 EXPOSED:                   T
```

Figure 12: An Image Histogram



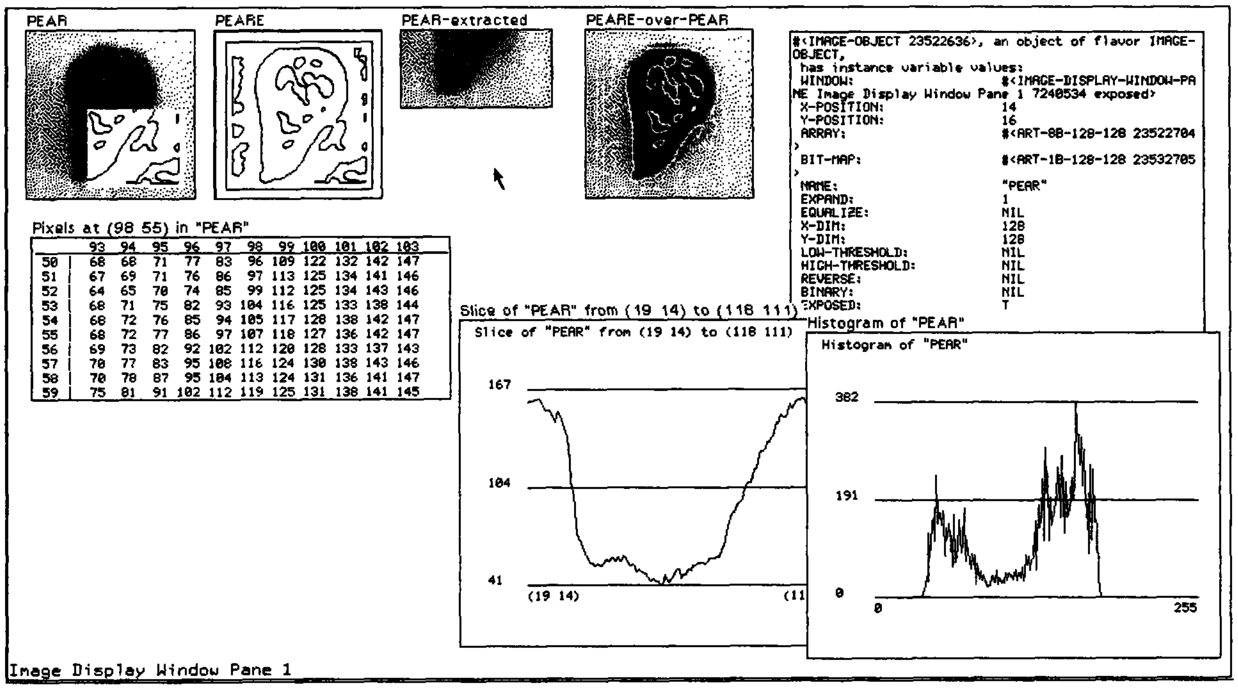
```
#<IMAGE-OBJECT 23522636>, an object of flavor IMAGE-
OBJECT,
 has instance variable values:
 WINDOW:                      #<IMAGE-DISPLAY-WINDOW-PA
NE Image Display Window Pane 1 7240534 exposed>
 X-POSITION:                  14
 Y-POSITION:                  16
 ARRAY:                       #<ART-8B-128-128 23522704
>
 BIT-MAP:                     #<ART-1B-128-128 23532705
>
 NAME:                        "PEAR"
 EXPAND:                      1
 EQUALIZE:                    NIL
 X-DIM:                       128
 Y-DIM:                       128
 LOW-THRESHOLD:               NIL
 HIGH-THRESHOLD:              NIL
 REVERSE:                     NIL
 BINARY:                      NIL
 EXPOSED:                     T
```

Pixels at (98 55) in "PEAR"

	93	94	95	96	97	98	99	100	101	102	103
50	68	68	71	77	83	96	109	122	132	142	147
51	67	69	71	76	86	97	113	125	134	141	146
52	64	65	70	74	85	99	112	125	134	143	146
53	68	71	75	82	93	104	116	125	133	138	144
54	68	72	76	85	94	105	117	128	138	142	147
55	68	72	77	86	97	107	118	127	136	142	147
56	69	73	82	92	102	112	120	128	133	137	143
57	70	77	83	95	108	116	124	130	138	143	146
58	70	78	87	95	104	113	124	131	136	141	147

```
Select one
                 PEARE
            PEAR-extracted
            PEARE-over-PEAR
                  NIL
      Pixels at (98 55) in "PEAR"
Slice of "PEAR" from (19 14) to (118 111)
          Histogram of "PEAR"
                 PEAR x
```

Figure 13: The Object List



```
#<IMAGE-OBJECT 23522636>, an object of flavor IMAGE-
OBJECT,
 has instance variable values:
 WINDOW:                      #<IMAGE-DISPLAY-WINDOW-PA
NE Image Display Window Pane 1 7240534 exposed>
 X-POSITION:                  14
 Y-POSITION:                  16
 ARRAY:                       #<ART-8B-128-128 23522704
>
 BIT-MAP:                     #<ART-1B-128-128 23532705
>
 NAME:                        "PEAR"
 EXPAND:                      1
 EQUALIZE:                    NIL
 X-DIM:                       128
 Y-DIM:                       128
 LOW-THRESHOLD:               NIL
 HIGH-THRESHOLD:              NIL
 REVERSE:                     NIL
 BINARY:                      NIL
 EXPOSED:                     T
```

Pixels at (98 55) in "PEAR"

	93	94	95	96	97	98	99	100	101	102	103
50	68	68	71	77	83	96	109	122	132	142	147
51	67	69	71	76	86	97	113	125	134	141	146
52	64	65	70	74	85	99	112	125	134	143	146
53	68	71	75	82	93	104	116	125	133	138	144
54	68	72	76	85	94	105	117	128	138	142	147
55	68	72	77	86	97	107	118	127	136	142	147
56	69	73	82	92	102	112	120	128	133	137	143
57	70	77	83	95	108	116	124	130	138	143	146
58	70	78	87	95	104	113	124	131	136	141	147
59	75	81	91	102	112	119	125	131	138	141	145

Figure 14: A Hole

3 Initializing the Display

To use GROK, load the file, and create an *image-display-window*, using one of the following functions, which create a window or a frame containing a pane of the appropriate flavor and a lisp listener pane. Also note that if GROK is called before a window has been created, it will ask you whether you want it to create one. GROK uses many global parameters to customize its behavior and a few global variables as general defaults. These globals are all of the form ***g-blat***, where *blat* is some descriptive name. The default window to which all output is directed is ***g-image-window***, which is set by **init-display** and **make-image-display-frame**. If you want multiple windows, you can create them using **make-image-display-window**, which returns the window. The image display functions take an *image-display-window* as an optional argument.

make-image-display-window *Function*

> Function **make-image-display-window** asks you to mouse corners of the window, creates the window, sets the default window ***g-image-window***, and returns an *image-display-window.*

make-image-display-frame &optional *fraction column-or-row lisp-on-top* *Function*

> Function **make-image-display-frame** constructs a frame with an *image-display-window* occupying *fraction* of the space, and the top (or bottom) part, selected by *lisp-on-top*, having a lisp listener associated with the frame. Each of the parameters has a default, which can be specified by a global parameter. *fraction* defaults to ***g-default-fraction***. *column-or-row* defaults to ***g-column-or-row***. *lisp-on-top* defaults to ***g-lisp-on-top***. **make-image-display-frame** sets the global variable ***g-image-display-frame*** and returns that as its value. To change the relative shape the panes, select **Edit Screen** from the system menu, using right click, edit the subwindows of the Image Display Listener, select *move multiple*, left click on the dividing edge in the frame, and use right-click to move the edge.

To bind a key, say circle, to select the *image display frame*, add the following line to your init file:

```
(v:select-grok-frame #/circle)
```

Typing SELECT-circle will make an *image display frame*, if there is none, otherwise it will select the frame. Creating a frame takes some time, so be prepared to wait.

4 Displaying Images

The display utility GROK requires an *image-display-window* for output, created in the fashion described above. When it displays an image, it creates an instance of the flavor *image-object*, one of whose slots is the image, and other slots contain the displayed bit-map and the display control parameters for that instance of the object. Once the object is in the window, it can be manipulated in many ways and redisplayed. The workhorse function at the bottom of all this is called **grok** (grey* -> greycrok -> grok) which knows about *image-display-windows* and frames and can dither images, both integer and floating point. Dithering is the process of creating a representation of an image on a display whose grey-scale resolution is less than that of the image. **grok** improves on the dithering algorithm of the previous image display routines by employing an algorithm selected from an article by Jarvis [CGIP, June 1976]. GROK treats an image as separate from its display, which is the bit-map produced by the dithering operation. For example, when you ask to redisplay an *image-object* with new parameters, the expansion parameter of that *image-object* is shown in the parameter slot in the menu. If it is not 1, for example 2, then changing it to 4 will create an *image-object* which is not 8 times the size of the original, but only 4. *image-objects* always use the original dimension and pixel values in the image and display them afresh, using the indicated parameters.

In addition, interaction with the image display routines now is facilitated by the use of more menus and mouse commands. Here the main display routine, **grok**, is described. In most cases, you'll want to use **show**, which has the same calling sequence, but inserts the symbol to which the array is bound as a default *image-object* name. **grok** and **show** have the same calling sequence. **grok** draws an *image-object* surrounded by a boundary of 1's (black on normal video), but you can remove this border by turning ***g-draw-borders*** off. **grok** can be ordered to *beep* when it is finished, by turning ***g-beep-when-done*** on. **grok** assumes reasonable defaults for images, but if ***g-use-last-input*** is set to t, **grok** will use the last display parameters for defaults. This is useful is displaying sequences of similar images with the same parameters. In this case, you might want to turn off the menu for parameter selection, by setting ***g-use-menu*** to **nil**.

grok &optional *array* &key *window equalize x-position y-position low-threshold high-threshold name expand reverse binary use-menu use-mouse* *Function*

Function **grok** creates and returns a display object for that image.

The following keyword argments can be specified:

:window An *image-display-window* for showing the image objects.

:equalize (**t** or **nil**) If equalize is **t**, the image is transformed, when displayed, so that the histogram of brightness values in the displayed image is uniform. Its default value is **nil**.

:x-position specifies a window x-coordinate for placement of upper left hand corner of the image.

:y-position specifies a window y-coordinate for placement of upper left hand corner of the image.

:low-threshold indicates a lower threshold for brightness values. It is also used to specify the threshold for binarizing images. It defaults to ***g-low-threshold***.

:high-threshold indicates the upper threshold for brightness values. It defaults to ***g-high-threshold***. Image values are truncated to lie between *low-threshold* and *high-threshold.*

:name provides a name for the image. It should be a string. It defaults to ***g-name***.

:expand takes on positive integer values. The image is scaled by *expand* before display. It defaults to 1.

:reverse reverses the video output for the image. Normally, low image values are displayed, in the dither routine, with a dot, which becomes black on the monitor. Depending on the contrast setting for the window, it may be necessary to use this option to get appropriate output. Binary images, such as edge maps, are shown with 1 mapped to black in standard video. Its default value is ***g-reverse***, which defaults to **'not-set**. When ***g-reverse*** is **'not-set**, the *reverse* option takes its cue from whether the screen on which the *image-display-window* resides is in black-on-white mode.

:binary specifies that the image is not to be dithered, and all values above the *low-threshold* are shown as 1. If the image is of type *art-1b*, it is treated as binary. When the *low-threshold* is not specified, the global ***g-binary-threshold*** is used for a threshold. The default value is **nil**. If ***g-reverse-binary-switch*** is **t**, a binary image will be displayed with reverse video.

:use-menu (**t** or **nil**) tells whether to use a choose-variable-values menu to enter display parameters. ***g-use-menu*** is the default.

:use-mouse (**t** or **nil**) tells whether to use the mouse to position the *image-object* in the window. The default is ***g-use-mouse***.

show is actually a macro front-end for **grok**, that is, it calls **grok** with a default name which is the symbol which is evaluated to provide the array for display.

Much of the functionality is in the menus. When, for example, a slice through the image or some image values are printed, they can be displayed in a temporary portion of the window, or into a permanent object. By pointing at an object in the window, and clicking, a menu of image operations is invoked. If the mouse is not on an object, the menu appears, and, after an operation is selected, if the operation needs a designated object, a list of object names appears. If none is selected, the operation is discontinued.

The mouse buttons are interpreted as follows:

Left-once Select the image-display window.

Left-twice Display the image pixel values in an area surrounding the mouse. Its default size is 11 x 11, but it varies with the magnitude of the pixel values. The default half-width of the area is ***g-nearby-values-size***. If the image contains floating point numbers, their values are printed in a field ***g-print-width*** wide. If this parameter is **nil**, as in the default, a menu will pop up and ask for an integer width for the field.

Middle-once Plot a section through the image, sliced along a line beginning at the current mouse-position, rubber-banded to the mouse. Another click fixes the line and causes the section to be plotted.

Middle-twice Plot a section through the image, sliced along a line through the current mouse-position, either horizontally or vertically, through the entire image. Another click tells whether it should be horizontal (Left) or vertical (Right).

Right-once Pop up the image display menu, and select an operation.

Right-twice System menu.

The action of the plotting and display functions can create permanent or temporary *image-objects* (*scratch-windows*) whose placement can be controlled by the mouse or not. Once a *scratch-window* is displayed, it remains until some input is directed to the window, either through the keyboard or by clicking the mouse over the window.

Following is a list of the operations on *image-objects*, accessible through right click:

De-expose The image-object remains but no longer appears in the window. It appears in the object list, and can be exposed, using the "Expose" option.

Describe Use the system **describe** command to list parameters of the object in a *scratch-window.*

Display with new parameters Invoke **grok** with the array of this object as its argument.

Expose Has the effect of displaying the *image-object* in the window at the last position it appeared and giving mouse control for further positioning.

Extract Use the mouse to describe a rectangle which is extracted from the array and redisplayed, using the display parameters of the object from which it has been extracted. The mouse specifies the upper left corner and the lower right corner of the rectangle, which can then be moved around in the *image-object.*

Halftone Create a halftone image of the selected *image-object* (using Dave Clemens' technique) and send it to the default screen hardcopy device. This process asks you for the size of the default half-tone dot (default 8).

Hardcopy Print a hardcopy of the selected *image-object* on the default screen hardcopy device.

Histogram Plot a histogram of the image intensities.

Hole In the same fashion as Extract allows you to position a rectangle in the *image-object*, then select an *image-object* from a menu. Then it shows the bit-map of the second object in the rectangle's position on the bit-map of the first.

Kill Get rid of the object once and for all. Bye-bye.

List objects Pop-up a menu with the names of the objects. **NOTE:** This same menu is used to reference objects by name, so use unique names.

Move Move the object under mouse control to a new position in the window. The handle in the object is in the center.

Overlay Overlay the selected object (A) upon an object selected by mouse-click (B) from the window. An ALU can be chosen from the menu (right-click) or by clicking left or middle to select ***g-default-alu-left*** or ***g-default-alu-middle***.

Refresh Bring the *image-object* to the front of the display, in the same position, and clean up any image junk on top of it.

Rename Alter the name of an object.

Refresh Window Clear the window and redisplays every *image-object.*

Flush all Objects Wait for a left click for confirmation, then kill all image objects, set the *image-count* to 1, move the default position to (***g-margin***,***g-margin***), and refresh the window.

5 Customizing GROK

GROK is flexible, and can be tailored to your wishes. Following is a list of the global variables you can set to modify the behavior of GROK, with default values:

g-default-fraction
Fraction of the frame occupied by images. Default: 0.7

g-column-or-row
Arrange panes in **make-image-display-frame** as a column (**t**) or row (**nil**). Default: **t**

g-lisp-on-top
Lisp is in top (or left) panel of frame. Default: **t**

g-use-mouse
Use the mouse to control placement of objects in the window. Default: **t**

g-use-menu
Use a choose-variables values menu to control **grok**. Default: **t**

g-use-last-input
Use the previous display parameters as defaults for **grok**. Default: **nil**

g-use-debugger
Errors in the image display process enter the debugger instead of just notifying the user. Default: **nil**

g-beep-when-done
At end of grokking, beep. Default: **nil**

g-default-alu-left
ALU for overlay. Default: **tv:alu-andca**

g-default-alu-middle
ALU for overlay. Default: **tv:alu-ior**

g-default-alu-string
Documentation on what to overlay.
Default: Select the object over which to lay - ALU = L: andca / M: ior / R: menu

g-draw-borders
Images have black borders. Default: **t**

g-draw-names
Images have names above them. Default: **t**

g-reverse-binary-switch
For binary images, put 1's out as black. Default: **t**

g-reverse
Reverse video. Default: **'not-set**

g-cleanup-image
Indicates that image-object is refreshed after slice or display pixels operation. Default: **t**

g-split-both-thresholds
If both thresholds are specified, then set the dither threshold at their mean. Avoids recomputing a histogram for the image. Default: **t**.

g-extract-bitbltable
Extracted sub-image forced to be bitbltable. Default: **nil**.

g-use-new-nearby
Allow you to move the nearby-values window around with the mouse. This defaults to **nil** only to preserve downward compatibility. Default: **nil**.

g-scratch-window-x
Sizes for initializing *scratch-windows*; should be a multiple of 32. Default: 320

g-scratch-window-y
Default: 250

g-graph-x-size
Default: 250 - should be approximately 70 less than ***g-scratch-window-x***.

g-graph-y-size
Default: 150 - should be approximately 100 less than ***g-scratch-window-y***.

g-nearby-values-size
Half-width of the displayed box, i.e., the box is $(1 + \ (* \ 2 \ * g - nearby - values - size*))$ wide. Default: 5.

g-cursor-box-edges
Borders of initial position of extraction cursor box. Default: '(50 50 100 100)

g-margin
Blank area on outside of *image-display-window*, in pixels. Default: 15

g-expand
Expansion factor. Default: 1

g-binary
Treat the image as binary. Default: **'not-set**

g-binary-threshold
Threshold for binarizing. Default: 0

g-low-threshold
Low-threshold. Default: **nil**

g-high-threshold
High-threshold. Default: **nil**

g-name
Default name for images. Default: **image-num**, where **num** counts the images in the window.

g-print-width
Width of output field for displaying floating point values in an image. Default: **nil.**

g-scratch-font
Font for print-out in the *scratch-window.* Default: **fonts:tvfont.**

g-font
Font for the *image-display-window.* Default: **fonts:hl10.**

g-dot-size
Size of halftone dot. Default: 8.

6 Other Utilities

GROK provides several functions for exporting data from its *image-objects.*

object-under-mouse &optional *image-window* *Function*

Function **object-under-mouse** waits for a mouse click on an *image-object* and returns the object under the mouse. The argument *image-window* defaults to ***g-image-window***.

image-under-mouse &optional *image-window* *Function*

Function **image-under-mouse** waits for a mouse click on an *image-object* and returns the image array of the object under the mouse. The argument *image-window* defaults to ***g-image-window***.

bitmap-under-mouse &optional *image-window* *Function*

Function **bitmap-under-mouse** waits for a mouse click on an *image-object* and returns the bit-map array of the object under the mouse. The argument *image-window* defaults to ***g-image-window***.

image-slice &optional *image-window* *Function*

Function **image-slice** waits for a mouse click on an *image-object* and then draws a rubber-band line showing the position of the section through the image it returns to you. The argument *image-window* defaults to ***g-image-window***.

Several (at least one) of the underlying functions used by GROK can be useful outside of the context of *image-display-windows.*

dither-image *image 1bit low-threshold high-threshold equalize expand reverse Function*

Function **dither-image** takes a multi-bit image *image* and an *art-1b* array *1bit* of the right size and produces a dithered representation of the image in the 1-bit array. The meanings of the parameters are the same as for **grok**.

equalizer *image low-threshold high-threshold* *Function*

Function **equalizer** takes a multi-bit image *image* and a low and high threshold (as used in **grok**), and returns multiple values *hist*, *map*, *lowest* and *factor*. *hist* is the histogram of the image values, scaled by (* *factor* (– *intensity lowest*)) to fit a maximum resolution of 256 buckets. *map* is a transfer array generated from the histogram, so that (*aref map* (* *factor* (– *intensity lowest*))) produces an equalized value.

histogram *image low-threshold high-threshold* *Function*

Function **histogram** takes a multi-bit image *image* and a low and high threshold (as used in **grok**), and returns multiple values *hist*, *lowest* and *factor*. *hist* is the histogram of the image values, scaled by (* *factor* (– *intensity lowest*)) to fit a maximum resolution of 256 buckets.

Acknowledgements

The assistance of Margaret Fleck, Harry Voorhees, Jim Mahoney, Patrick O'Donnell, Dave Braunegg, and Philippe Brou is gratefully acknowledged. GROK began with Margaret Fleck's GREYCROK and grew by stealing ideas from all the above, and code in particular from Harry Voorhees, whose utilities have been incorporated into GROK. Had Symbolics written a clearer window manual life would have been easier. Carol Bonomo had nothing to do with this.