Semper 6 *Plus*

QUICK REFERENCE LIST

Synoptics

Trademarks:

Semper is a registered trademark of *Synoptics Limited*
IBM PC is a trademark of International Business Machines Inc
Hewlett-Packard LaserJet+ is a registered trademark of Hewlett-Packard
PostScript is a registered trademark of Adobe Systems Incorporated
Quantimet is a registered trademark of Cambridge Instruments
Silicon Graphics Personal IRIS 4D is a trademark of Silicon Graphics Inc
Sun is a registered trademark of Sun Microsystems Inc
VAX is a registered trademark of Digital Equipment Corporation

Manual number: SEM011
Update number: 0
Printed: June 1990

Table OF CONTENTS

Chapter 1: Introduction

About this manual 1-1
How it's organised 1-1
What else to read? 1-1
A note about syntax 1-2

Chapter 2: Semper 6 Commands

Overview 2-1
Variables and terminal information 2-3
Program execution 2-6
Picture display and inspection 2-10
Picture management 2-14
Device management 2-17
Point by point operations 2-19
Geometric operations 2-21
Local operations 2-23
Transformation and filtering 2-25
Line, lattice and rotational averaging 2-27
Correlation functions and alignment 2-29
Particle analysis 2-30
Morphology 2-33
Remote sensing 2-34
User interface creation 2-36
Miscellaneous 2-38

Chapter 3: Installation Specific Commands

Overview 3-1
Framestore configuration and control 3-3
Image acquisition 3-5
File reading/writing 3-7
Miscellaneous 3-9

Index I-1

Chapter 1

INTRODUCTION

About this manual

This manual provides a brief survey of all Semper 6 commands. You may find it useful for quick reference when you are not at a computer terminal or for identifying the name by which Semper refers to a particular operation you may wish to perform.

How it's organised

Each command is listed in *Chapter 2, Semper 6 Commands* with an example of its use and a short description of its function. The commands are grouped according to function, for example, **analyse** is grouped under the heading *Particle Analysis*; **erode** can be found under the heading *Morphology*. If you know the name of a command, but are unsure of which heading it falls under, refer to the *Index*, which lists all commands in alphabetical order.

Chapter 3 provides a list of installation specific commands, grouped according to function. These commands are placed in a separate chapter as they apply only to particular hardware combinations. For example image acquisition commands such as **animate** and **live** are available only with certain machine/framestore combinations.

What else to read?

If you need further information about a command, refer to the manual:

Semper 6 Command Reference

This provides a complete description of each command, giving its full syntax and many illustrations of its use.

A further document:

Semper 6 Command Summary

provides a one-line description of each Semper command and is useful as a memory aid during a Semper session.

Comprehensive help facilities are also available on-line during a Semper session. Type **help** at the terminal for details. You can also use the **syntax** command to check the keys and options of a command, for example:

```
syntax ladjust
```

provides information about the **ladjust** command.

A note about syntax

All parts of a Semper command can be abbreviated to three letters, for example:

```
lad ran 64,128 sat
```

means the same to Semper as:

```
ladjust range 64,128 saturation
```

The exception is if you force an option off using **no**. For example, the command **survey noverify** can be abbreviated to:

```
sur nover
```

but *not* to:

```
sur nov
```

For clarity, all commands are shown in their unabbreviated form in this manual.

You can assign options and keys new global defaults by an assignment, for example:

```
verify=yes    erase=no    radius=25
```

and can still override these defaults by local use in individual commands, for example:

```
display 2 erase
```

You can continue a command line over several input lines by adding a plus sign (+) at the end of any line to be continued. For example:

```
ladjust range +
64,128 saturation
```

is equivalent to the single line:

```
ladjust range 64,128 saturation
```

Note that a *single* typed line must not exceed 80 characters and that a command line that consists of several lines must not exceed the size of the command line buffer (type **show system** to see the size for your installation).

For a full discussion of command syntax, refer to *Chapter 2, The Command Interpreter* in the following manual:

Advanced Users' Guide

contained in the *Semper 6 Guide.*

Chapter 2

SEMPER 6 COMMANDS

Overview

This chapter provides a description and examples of Semper 6 commands. The commands are grouped according to function. This allows you to isolate the commands that are suitable for a particular task, for example, particle analysis. Figure 2-1 overleaf, illustrates the functions of Semper commands.

A list of function headings, in the order that they appear in this chapter, is given below:

- Variables and terminal information
- Program execution
- Picture display and inspection
- Picture management
- Device management
- Point by point operations
- Geometric operations
- Local operations
- Transformation and filtering
- Line, lattice and rotational averaging
- Correlation functions and alignment
- Particle analysis
- Morphology
- User interface creation
- Remote sensing
- Miscellaneous

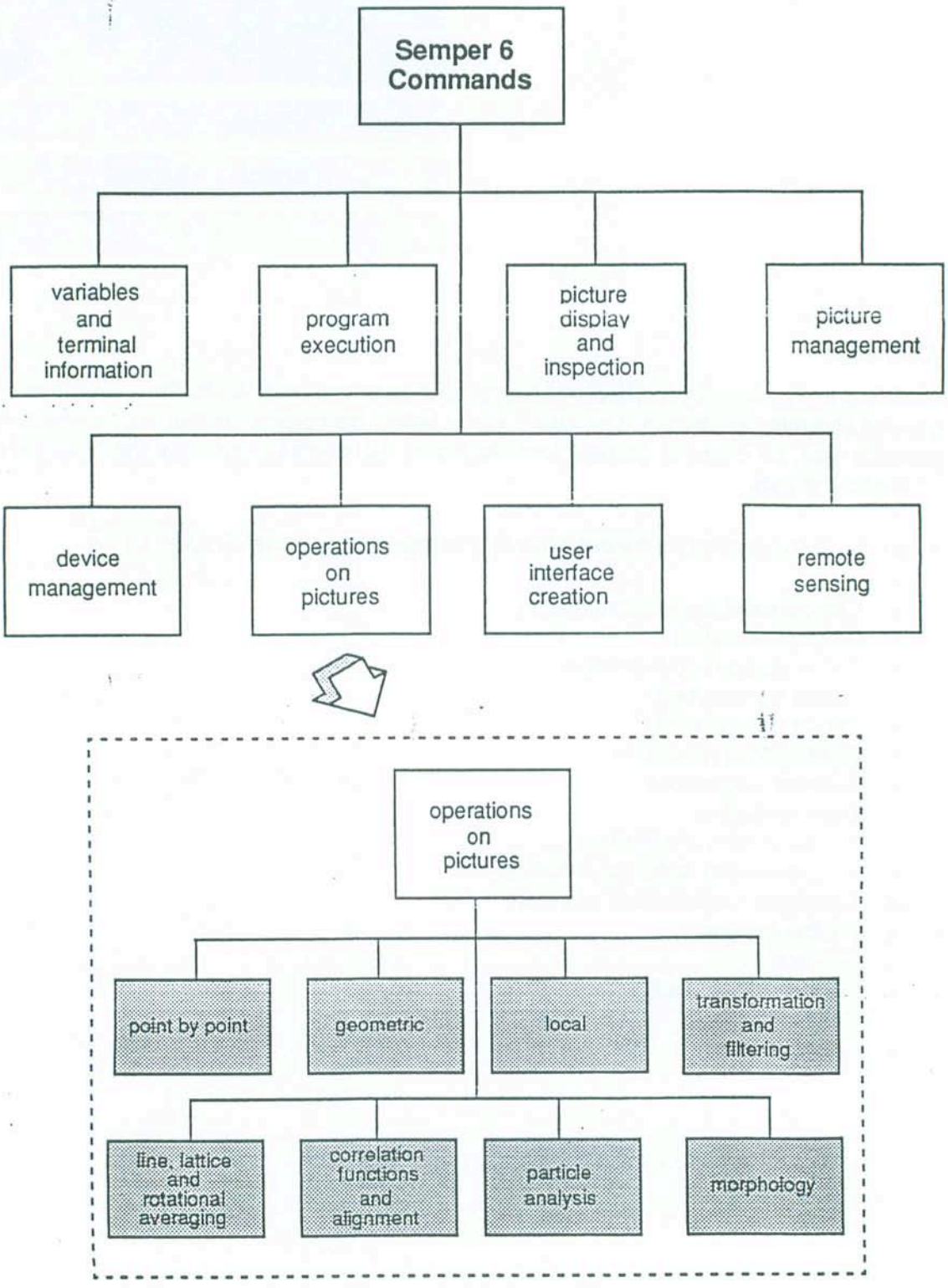

Figure 2-1. Semper 6 Commands By Function

variables and terminal information

name = val sets the named variable to the specified value.

```
x = 30
```

ask types a prompt on the terminal and accepts numerical values (which may be a general expression) for named variables in turn. Note that **ask** can only be used in programs or macros.

```
ask x,y,z                                    (sets the variables x, y and z)
ask 'Drift for picture ',np, ': ' drift      (sets the variable drift)
unless set(p) & set(p2) ask 'First, last pictures: 'p,p2 (sets p, p2)
```

beep, bell or **buzzer** sound the bell on the terminal.

```
beep
```

cls clears the terminal screen and puts the cursor at the home position. It gives you greater control over how information is presented on the terminal.

```
cls
```

echo controls the mapping of Semper's six logical output streams to the terminal or to any log files that you have assigned (use the command **show echo** to list the current echo settings).

```
echo terminal none console      (only console output to terminal)
echo device 2 all noconsole     (sends everything but console output to device
                                 2, which is the log file)
```

help provides on-line information for a command or topic.

```
help                            (provides details of available help information)
help devices                    (lists help information for the topic devices)
help/full fourier ps            (lists full help entry for commands fourier and ps)
help/topics a b c               (lists help entries under a b c)
help/log                        (outputs entire help information to log output stream)
```

inkey waits for one or more terminal keys to be pressed and sets each of the variables listed in the command to the internal character code for each key pressed in turn, that is, to the decimal value of the ASCII code for each key. If no variable names are given, **inkey** waits for any single key press.

```
inkey
inkey k1,k2,k3; type k1,k2,k3
```

local specifies the names of variables to be restored to their previous or current state when the command line or program containing the **local** command terminates. The variables are restored to the state they were in before the **local** command was executed. Note that loop variables are automatically restored when loops terminate, that is, they are local to their for loop.

```
local x,y; ...
local ncols, nrows, nlays, class, form; pcb
type ncols, nrows, nlays, class, form
```

log prints text and/or numerical values to the log output stream. The output takes exactly the same form as in the **type** command.

```
log 'The result is ', 1/n*root(pi)
log 'Subregion mean and sd: ', mean, sd
```

p or **pixel** sets the value of one or more pixels in the current picture. (Note that the *function p(x,y,z)* provides the reverse facility, that is, it returns the value of a specified pixel).

```
select 4; p 0,0 = 0
p 2,-2 = .2,1,.2,.5,.2,.1        (sets 5 pixels rightwards from 2,–2)
```

page controls the size of the terminal window and how output to it is formatted. Output to the terminal may be truncated to a particular width or it may wrap-around. It may also be paged, so that you can read it in your own time. Use the command **show page** to list all the current page settings.

```
page nowrap prompt              (truncates long lines and enables page prompt)
page width 40 length 12         (defines new terminal window size)
page aspect 2.33                (sets up character aspect ratio)
```

pointer defines the gearing and sensitivity for the pointing device (the mouse), that is, how the cursor responds to mouse movements.

```
pointer gearing 4 sensitivity 2
pointer gearing 10,1            (sets different gearing in x and y directions)
```

report reports the last error or trap condition that was met.

```
report trap
report error
```

show types information on the current state of the system. The command **show** by itself lists the available options. Use the **show** command to discover:

• how Semper is configured	`(show system)`
• the error messages generated by Semper	`(show errors)`
• available Semper commands	`(show commands)`
• the value of all variables set	`(show variables)`
• the current echo and page settings	`(show echo and show page)`
• factorisable sizes, up to the maximum row length	`(show sizes)`
• the devices assigned	`(show devices)`
• the defined display partitions	`(show partitions`
• the defined look-up tables	`(show luts)`
• the available programs	`(show programs)`
• the defined named macros	`(show macros)`
• the time of day	`(show time)`

syntax types details of the keys and options for a given command name (use the command **show commands** to obtain a list of all the command names).

```
syntax view
syntax mask
```

time types the time that has elapsed in seconds since the start of the session or since the last **time reset** command.

```
time
time reset; ...; time          (measure time to execute commands)
```

type types text and/or numerical values (may be general expressions) on the console. If several items are to be typed, they must be separated by commas.

```
type 'The mean value is ', sum/number
type 'Central pixel is ', p(0,0), ' and squared mean ', mean^2
```

unset unsets variables previously set by an assignment. (Use the function *set(name)* to test whether a particular variable is currently set).

```
unset mark,preset
```

program

execution

@name is equivalent to the text of the macro name (**show macros** lists all named macros).

```
xwires; mark @xy text 'Obj A'          (@xy = position x,y)
```

@number is equivalent to the text stored in the numbered macro, defined previously using the **macro** command.

```
macro 49; xwires region; survey @region full noverify
type sd/mean
for i=1,5; @49
```

add loads a new program or replaces an existing program in a program library. You can also use **add** to convert a numbered macro into a program.

```
add program 'myprog'                    (you are then prompted for program input)
add name 'sort.spl'                     (load sort.spl into the current program library)
add macro 2:90 program 'macro90'        (converts the macro in picture 90 on
                                         device 2 into a program called macro90
                                         on the current program library)
add name 'myprogs' program 'prog2'      (reads the program named prog2 from
                                         the file myprogs.spl into the current
                                         program library)
```

break terminates the corresponding **for** loop, that is, the command after the terminating **loop** command is executed. An optional loop variable name specifies which loop to terminate (default is innermost active loop).

```
for i=1,5; ...; break; ...; loop; ...
for x=0,1,0.1; for y=0,1,0.1;...;break x;...; loop y; loop x;...
```

edit runs a text editor that enables you to alter the text of a numbered macro or to create a new one.

```
edit 4                                  (w to quit, ? for help)
edit to 901                             (creates new macro with editor)
```

end should be the last command in a program or run file. It is treated as a **stop** command if entered directly from the terminal.

```
end
```

exit terminates your session (as do the commands **quit, stop** and **end**).

```
exit
```

for makes Semper loop over subsequent commands up to the corresponding **loop** command. The loop may count in either a positive or negative direction by any specified step. Loops may be nested. See **loop, next** and **break** for more detail about controlling loops.

```
for n 1,5; display n; loop
for x=2,0,.3; type x; loop x
for x=-5,5; for y=-9,9; p x,y=max; loop y; loop x
```

if, unless makes execution of the rest of the command conditional on a logical value (which may be a general expression), immediately following **if** or **unless**.

```
if n>10 display n
unless err<.1 | ncycles=10 jump iterate
if set(rbg) unless abs(max-min)<40 fit subtract
```

jump makes Semper branch to the command prefixed with the specified label. A label is a name followed by a colon. In interactive use, the label must appear in the same command line as the **jump** command.

```
n=1; next: display n; n=n+1; unless n>10 jump next
pcb; if class=3 jump on; type 'Picture must be Fourier';
+return;on:
```

library runs the specified library program. Programs are stored in disc files called program libraries (the command **show programs** lists all currently available programs). Program names must be entered in full (they cannot be abbreviated to three letters).

```
library partitions
library molecule
```

list types the text of a program or numbered macro at the console.

```
list 35
list program 'molecule'
list all
```

loop is the terminating statement for a **for** loop, that is, it increments the loop variable by the specified step and if the variable does not exceed the end value, it executes a jump to the command immediately following the corresponding **for** command. The loop variable name may optionally be included to enforce checking for proper nesting of **for** loops.

```
for i=1,5; ...; loop
for x=0,1,0.1; ...; for y=0,1,0.1; ...; loop y; loop x
```

macro defines a numbered macro, a sequence of Semper commands stored as a class *Macro* picture for later execution via the construction ***@number.***

```
macro 89; xwires; extract size 128 @xy to 51; ps ln; survey; +
min=mean; display preset
@89
```

next terminates the current iteration of the corresponding **for** loop, that is, it executes the terminating **loop** command. An optional loop variable name specifies which **loop** command to execute (default is the **loop** command terminating the innermost active loop).

```
for i=1,5; ...; next; ...; loop
for x=0,1,0.1; ...; for y=0,1,0.1;...;next x; ...;loop y; loop x
```

order controls the way in which program libraries are searched when a program is invoked. The list of device numbers you specify are placed first in the search order of program libraries. The default order is to search the last assigned program library first. The command by itself lists the current search order.

```
order
order 3,4
order 2,5,3 noverify
```

quit terminates your session (as do the commands **stop**, **exit** and **end**).

```
quit
```

rename renames a program in a program library.

```
rename program 'oldprog' as 'newprog'
```

return returns control from a program or run file (see **library** and **run**).

```
return
```

run executes the series of commands contained in the specified text file. The **run** command terminates if **return**, **end** or *end-of-file* is encountered. The last command in the file should be **end**. The **run** command itself may not be used in any run file.

```
run name 'commands.run'
```

stop terminates your session (as do the commands **quit**, **exit** and **end**).

```
stop
```

wait suspends execution either until you press a key or mouse button (it outputs a prompt to this effect), or else it waits for a stated time in seconds.

```
wait
wait 5
```

Picture display and inspection

contour draws contour maps of pictures or subregions on the display overlay, with optional interpolated magnification to achieve smooth presentation of subregions.

```
contour 1 times 4
contour 91 size 100 top left times 2 levels 3
```

display produces grey-level displays of pictures or subregions, graphs of 1-D pictures and histograms, and character-form displays on the terminal and to the log output stream. Automatic grey-scaling is performed, unless you quote the option **preset**. The output of most commands can in fact be sent directly to the display, for example, **sharpen 1 to display**, provided that you set the variables *min* and *max* suitably first to define the black and white levels used for display scaling.

```
display 1
display 67 to display:3 noerase
display type width 150
display 21 size 100 top left negated times 2
display 2 noscale
min=50 max=200 display 23 display:2 preset
```

examine types details of picture size, class, form, title, etc.

```
examine full                        (current picture, with supplementary details)
examine display:5,display:10        (any pictures in range display:5 to display:10)
examine all                         (all pictures on current device cd)
examine device 4                    (all pictures on device 4)
```

histogram creates an intensity histogram of a picture or subregion, producing a class *Histogram* picture.

```
histogram 73
xwires region; histogram @region to 50; display
histogram channels 50 to 2
```

ladjust lets you adjust the current display look-up table interactively, using a mouse or cursor keys. The interaction may be through the keyboard or the mouse or both. Options allow you to change the hue, saturation, brightness and contrast. Additionally, **ladjust** can be used to highlight a particular range of output levels.

```
ladjust brightness contrast    (alters the brightness and contrast)
ladjust range 64,128 hue sat   (alters hue and saturation in the given range)
```

lset lets you set the contents of the current display look-up table. A particular range of output levels may be set using the **range** keys.

```
lset brightness 0,1            (sets the look-up table to a linear monochrome ramp)
lset brightness 1 hue 120 sat 0,1 (sets the look-up table to shades of green)
```

lut controls the contents of the display look-up tables, where these exist, allowing you to generate new tables or adjust existing ones (optionally, interactively via terminal keys).

```
lut 1 reset; view frame
lut invert
lut highlight 80,120 keys
lut 2 create colour; view lut 2
lut 3 create false
```

mark marks positions, subregions, text and position lists on the display.

```
mark partition 5 border                (outlines the partition)
mark size 200,300 bottom right         (marks rectangular subregion)
mark radius 150 @xy                    (marks circle)
mark @xy mkmode 3 mksize 5             (marks 11 point square box)
mark text 'kernel size ',n below       (marks text below picture, as title)
mark @xy rj text 'xcf peak ->'         (marks right justified text)
mark display:8 with 31                 (marks position list 31 on display:8)
```

postscript creates a *PostScript* file of a picture or region of the display. When creating a *PostScript* file from the display it will usually include any annotation (if the display supports the reading back of data from the overlay). There is an option which allows *encapsulated PostScript* output – a form suitable for inclusion into other documents. It is possible for the *Postscript* output to be sent directly to a printer, provided that the host operating system allows you to refer to an output device as if it were a file.

```
postscript 1 name 'image'
postscript partition 4 name 'all'
postscript 12 encapsulated name 'image'
postscript display landscape name 'laser:' old
```

print prints the pixel values of a small subregion of a picture to the console.

```
print 1
xwires; print @xy size 5
print 16 top left mp                    (prints modulus and phase)
```

sheet treats pixels as height values defining a surface in 3-D, and generates a shaded image of the surface as viewed from a specified direction.

```
min=0 max=255 sheet                     (outputs to the display)
sheet to 51 range 10,80 border 0        (scales heights and adds border)
sheet theta -pi/3                       (view in a different direction)
```

solid thresholds a 3-D picture to obtain the description of a solid object, and generates a shaded image of it as viewed from a specified direction.

```
solid 1 threshold .05                   (outputs to the display)
solid to 51 theta -pi/4                 (view in a different direction)
```

spc (superpose contours) marks contour lines in an image, resetting the pixels themselves rather than drawing lines in the display overlay (see **contour**). By default, contours are marked bright where the picture is dark and vice versa.

```
survey; spc to display levels 11
spc 1 2 ln range 5,1000                 (log-spaced contours, from 5 to 1000)
```

survey scans pictures or subregions and stores the pixel range in variables *min* and *max*, the mean in *mean* (and *me2* if complex), and the rms modulus deviation from the mean in *sd*.

```
survey 53 noverify                      (only min and max set)
xwires region; survey @region full      (min, max, mean and sd set)
```

view selects a display frame, partition or picture for viewing on the monitor screen, setting the zoom factor and look-up table used (which controls whether presentation is in monochrome, false colour or full colour form). (You can establish a default lut number for each partition using the **partition** command).

```
view display:3 zoom 8 @xy       (picture position x,y is centred on screen)
view display:21,display:22 zoom 4 (alternates between two display pictures)
lut 3 create false; view frame 2 lut 3 (views frame 2 in false colour mode)
```

xwires allows you to enter positions, lines, subregions, circles, arcs and curves directly onto the display using the cursor.

```
xwires; type x,y                          (sets variables x and y)
xwires line; project angle theta          (sets x, y, r and theta)
xwires region; cut @region to 51          (sets x, y, r and r2)
xwires curve to 50                  (records curve as class Plist picture)
xwires graph to 50                        (records graph as 1-D picture)
xwires section;extract @section to 51     (takes 1-D section of a picture)
```

ymod displays pictures or subregions as a perspective plot of a surface, with height proportional to brightness.

```
ymod 48 preset
ymod size 32 times 4
```

Picture management

copy copies one or more pictures from place to place, within or between devices. It is the only command able to create tape pictures. **copy** can also be used to copy a program or to change the format of a picture.

```
copy 1 to 2                                  (copies picture 1 to picture 2)
copy 11,14 to 21                             (copies the pictures 11, 12, 13 and 14 to
                                              the pictures 21, 22, 23 and 24)
copy program 'myprog' as 'newprog'           (copies the program called myprog.spl
                                              to the file newprog.spl)
```

create allocates space for a new picture, optionally filling it with a constant value. You can also use it to re-declare the dimensions etc. of display pictures.

```
create 3 integer size 500,200 value 0; paste 1 left
paste 2 right
create 5 size 100,1; origin left; calculate a^2/(a^2+x^2)
display 1; create display size 300; copy display to 2
```

delete eliminates pictures that are no longer needed, making their space available for other pictures. You can also use **delete** to delete a program from a program library.

```
delete 4:51
delete 200,300                    (deletes all pictures in the range 200 to 300 inclusive)
delete program 'new'
```

letter adds lettering at the bottom of a picture or creates a new picture containing nothing but text.

```
letter 1 text 'after background levelling'
letter 3 title                                (takes text from picture title)
letter to 11 text 'Band 2'; paste to 3 top right
```

origin defines a new coordinate origin for a picture.

```
origin 24 bottom left
xwires; origin 30 @xy
origin reset
```

pcb produces picture control block information for a picture by setting the variables *ncols*, *nrows* and *nlays* to the picture dimensions, *x1*, *x2*, *y1*, *y2*, *z1* and *z2* to the coordinate ranges, *class* to the class number, *form* to the form number and *wpf* (write protection flag) to yes or no .

```
pcb 30; unless class=3 fourier
pcb; for x=x1,x2; p x=sin(pi*x^2); loop
```

read reads a picture from an ordinary formatted or unformatted file. The file may have been output from a program or package of your own. While not very fast, the **read** command provides access to pictures coming from a wide variety of sources. You should be able to read formatted files output by the Semper **write** command on any system).

```
read 4 name 'meltspin.lis'
read 1 name 'dump.dat' unformatted
```

reclass changes the class recorded for a picture, forcing Semper to treat the data differently later, for example, to Fourier transform a power spectrum (class *Spectrum*) you must first change its class to *Image*.

```
reclass 30 image
```

renumber changes the number identifying a disc picture, without copying or altering the contents.

```
renumber 10 51
```

save dumps one or more pictures to a newly created file with the same format as a normal Semper picture disc file. You use the **save** command for backup or archive purposes. This is the simplest and most efficient way of saving pictures.

```
save 30,40 name 'emsa.sav'
save display:1,display:10                (prompts for a file name)
```

select sets the protected variable *select* to the given picture number, making the corresponding picture current. The default picture number for many of the processing commands is taken from *select*.

```
select 15; fourier
```

title replaces or adds to the title recorded with a picture.

```
title text 'Frame ',n,' with background levelled'
title 20 add text '- ln spectrum'
```

wp sets or unsets write protection for individual pictures.

```
wp 35 off; delete
wp 30,50                    (write-protects any picture in the range 30 to 50)
```

write writes a picture to a dynamically created file. While much slower than the command **save**, it allows you to write pictures in a form accessible to other programs or packages of your own. You can use the command **read** to recover pictures that have been output with **write**.

```
write 4 name 'meltspin.lis'
write 1 name 'dump.dat' unformatted
write display name 'result',n format '(10F8.4)'
```

Device management

assign assigns a device (display, picture disc, log file, program library, help library or tape) to a Semper session. An unused device number is assigned and returned in the variable *n*. Tapes are mounted if necessary.

```
assign display
assign file name 'semper.log'          (assign file for logging)
assign name 'semper.dat' wp            (entire device to be write-protected)
assign new name 'protem' size 300      (creates new file, 300kB in size)
assign help name 'mylib'               (assign a help file)
assign tape                            (Semper prompts for name)
assign scratch program size 20         (creates a temporary program library)
```

close is only relevant to installations where display output cannot be viewed directly. The **close** command causes any buffered display output to be sent to a hard-copy device.

```
close
```

compress condenses a disc device, grouping empty segments together. Use it if Semper returns the message '*Disc fragmented*' when trying to create a new picture.

```
compress device 3
```

deassign deassigns a device, releasing the device number and the physical resource. Tapes are dismounted if appropriate.

```
deassign                               (deassigns device cd)
deassign display                       (deassigns display device)
deassign device 4 delete               (deletes file as well)
```

directory prints details of a disc device directory on the terminal, including the amount of free space remaining in the device.

```
directory
directory device 3
```

erase erases a display picture, partition or frame (or a subregion of these). You can erase the image and overlay separately.

```
erase display                          (erases all of current display picture)
erase size 200 top left                (erases picture subregion)
erase partition display:2 overlay      (erases partition overlay only)
erase frame 2 image                    (erases image stored in frame 2)
xwires region;erase @region            (erases part of a picture)
```

flush forces any disc areas buffered in memory to be copied back to the physical disc immediately, avoiding any data loss in the case of a computer failure.

```
flush
```

partition creates or redefines a display partition on a given frame and with a given size and position. It also establishes a default lut number for viewing the partition.

```
partition 3 size 256 top left              (default lut number is 1)
partition 21 size 128,512 right frame 2 lut 2
partition 5 size 128 position 64,0     (displaced 64,0 from frame centre)
xwires frame region;partition 4 @region  (defines a partition with cursor)
partition size 256 frame 1,3               (defines partition with 3 frames)
partition 12 enquire                       (returns partition size/position)
```

ramps fills a display picture, partition or frame (or a subregion of one) with a simple grey scale ramp pattern, to allow adjustment of monitor brightness and contrast settings or to examine the details of the lut in force.

```
ramps frame                            (fills the whole of frame 1)
ramps partition 5                      (creates a ramps pattern in partition 5)
ramps full                             (ramp with unit intensity increments)
```

reinitialise re-initialises a disc device with a fresh directory, wiping out any existing data completely irrespective of any write-protect flags on individual pictures (Semper asks you to confirm the command before proceeding).

```
reinitialise device 2 slots 100
```

rewind rewinds a tape device. Tapes are rewound automatically when necessary, but it can save time later to start a tape rewinding while you do other things.

```
rewind device 5
```

Point by Point

operations

calculate calculates any combination of pixel values that can be specified in terms of a general Semper expression. The expression is evaluated one pixel position at a time, using the corresponding pixel values in each of the given source pictures. The source picture numbers must include a colon for them to be recognised as such. The names *x*, *y*, z and *rr* cause the relevant coordinate offset and distance squared from the picture origin to be substituted in expressions.

```
calculate :2-:3 to :4                    (difference of pictures 2 and 3)
calculate max(:2,:3) to display          (brighter pixels of pictures 2 and 3)
survey; calculate (:sel-mean)/sd         (normalises to zero mean, unit sd)
calculate root(:50)-ln(:51/2)            (calculates square root and log)
calculate :select>thresh                 (1 if pixel>thresh, otherwise 0)
calculate :1*sin(x-.6*y) to display (imposes sinusoidal grating on picture 1)
```

correct forces the modulus of a complex picture to match that of a specified reference picture, but preserves the phase – a common step in iterative phase-retrieval algorithms.

```
correct 20 with 5
```

fit fits a planar ramp to a picture by the method of least squares, and subtracts or divides by the result, so as to correct for uneven background and signal levels in a picture. The ramps can be subtracted without its constant term (its value at the origin), preserving the background level at the origin.

```
fit display subtract noconstant  (subtracts fitted ramp: a*x + b*y)
fit divide fp                    (divides by fitted ramp: a*x + b*y + c)
```

gaussian creates a picture with a single real 2-D or 3-D Gaussian profile peak at its centre.

```
gaussian 90 size 256 radius 10
```

lorentzian creates a picture with a single real 2-D or 3-D Lorentzian profile peak at its centre.

```
lorentzian 10 size 200 radius 8
```

map passes a picture through an arbitrary intensity map defined in a 1-D picture containing map function sample values for a stated range. It also performs histogram equalisation if a histogram is provided instead of a map.

```
map 20 range 0,255                    (with map assumed to be in picture 999)
map with 50                           (range of min and max assumed)
histogram 3 size 100 to 51; map 3 with 51 gaussian
```

mask suppresses brightness variations inside or outside a given circular or polygonal subregion of a picture. Circular masks may be soft-edged.

```
mask inside radius 85 @xy             (circular mask centred at x,y)
mask display width 10                 (soft-edged mask about 10 pixels wide)
xwires curve to 52; mask 40 inside with 52 value 0
```

negate re-scales a picture so as to reverse its grey scale, interchanging its minimum and maximum intensity values.

```
negate
negate 23 preset                      (reverses current min and max values)
```

noise generates a noise-limited version of a picture, for test purposes. The noise is uncorrelated from point to point.

```
noise 15 dose 8                       (Poisson distributed, with 8 quanta per pixel)
noise width 23                        (additive Gaussian noise, with sd of 23)
```

scale re-scales a picture linearly to a stated new range or mean and standard deviation.

```
scale 101 to 3                        (default range is 0, 255)
scale 10 range 30,220
scale msd 1,.1                        (scales to mean of 1 and sd of 0.1)
```

Geometrical operations

cut cuts out a subregion of a picture (with unit sampling and no rotation). Use **paste** to insert a cut region into a picture.

```
cut to 40 size 200 top
xwires region; cut @region
cut layers 2,4 to 20                    (subregion three layers thick)
```

extract extracts any region of a picture. Non-unit sampling may be used and the region may also be rotated or skewed. The parts of the region that extend beyond the source picture simply 'wrap around', that is, as if the source picture repeats itself in all directions. Interpolation is used if sampling points do not fall exactly on individual source pixels.

```
extract size 120
extract 1 2 size 64 sampling .53 @xy angle pi/4
u=1.,2 v=.15,.9 extract size 50 uv mark display
xwires section; extract @section to 51    (extracts straight section)
xwires curve; extract with 999 to 51 (extracts section along arbitrary curve)
```

find finds the highest pixel, lowest pixel, or 'centre of mass' pixel, in a picture or circular subregion.

```
find; type x,y
find lowest radius 50 @xy
find cm
```

fullplane converts a right half-plane *Fourier* or *Spectrum* class picture into its full-plane equivalent, by rotating the right half (conjugated if complex) to the left.

```
ps 23; fullplane
survey 23; fullplane to display
```

halfplane converts a full-plane *Fourier* or *Spectrum* class picture into its right half-plane equivalent, by discarding the left half.

```
halfplane 1 to 2
```

magnify magnifies pictures or subregions by integer factors, either interpolating or repeating pixels.

```
magnify 21 times 4
xwires region; magnify @region to 23; spc
magnify to 387 size 50 replicating
```

paste inserts one picture into another, truncating anything that overflows the output picture.

```
paste 10 to 128 top right
paste 50 @xy                                (inserts into the current picture)
paste 4:2 layer 2
```

peaks finds locals peaks in a picture and records their positions in a *Plist* class picture.

```
peaks 1 threshold .75                       (to picture 999 in default)
peaks threshold 2.5 radius 5 to 52  (records local c.m. around each peak)
```

rotate rotates a square picture of a factorisable size efficiently through large angles.

```
rotate 87 88 angle pi/3
```

separate separates one or more layers from a multi-layer picture as a series of single-layer pictures.

```
separate 1 to 51                    (separates all layers as pictures 51,52..)
separate 1 to 51 layer 3
separate to 11 layers 2,6           (produces pictures 11 to 15)
```

stack combines several pictures (with identical layer sizes) into a single multi-layer picture.

```
stack 51,55 to 1
stack 51,55 to 1 layers 2,3         (layers 2 and 3 of each source picture)
```

transpose transposes a square picture of a factorisable size (interchanges rows and columns).

```
transpose 65 to 23
transpose display
```

turn performs various simple reflections and rotations of pictures (image sizes must be square and a power of two).

```
turn 63 over
turn upsidedown
```

Local operations

differentiate obtains the intensity gradient of a picture in any direction, using a three-point operator.

```
differentiate 25 73 angle pi/3
```

edge applies an gradient-magnitude or largest diagonal difference (*Roberts*) edge detection operator.

```
edge 1 to 3
edge 1 roberts
```

fir applies a *Fir* (*Finite Impulse Response*) filter, which is a local weighted average, to a picture, with a nearly arbitrary kernel. *Gaussian* and *Laplacean* kernels are built-in, and relatively efficient code is provided for 3x3 and 5x5 kernels.

```
fir 1 gaussian                          (smooths with 5x5 gaussian operator)
fir 2 to 4 subtract laplacean           (sharpens by subtracting a laplacean
                                         kernel)

create 90 size 3; ...+
...p -1,1=1,2,1; p -1,0=2,4,2; p -1,-1=1,2,1; print;
fir display with 90                     (define a 3x3 kernel.. and apply it)
```

hp applies a high-pass filter to a picture, subtracting from each pixel the local mean over an arbitrarily large square block.

```
hp over 50                                      (over 50x50 block)
min=-30 max=30 hp display:2 to display:5        (over 5x5 block)
```

lmean calculates for each pixel the local mean over an arbitrarily large square block.

```
lmean display                                   (over 5x5 block)
lmean display to 3 over 50                      (over 50x50 block)
```

lsd calculates for each pixel the standard deviation over an arbitrarily large block. It is quite a good edge-detection operator.

```
hp 1 over 20; lsd over 20 to 2; calc :1/:2   (standardise local mean and
                                              standard deviation)
```

lvariance calculates for each pixel the local mean square over an arbitrarily large square block.

```
lvariance 2 over 50                  (over 50x50 block)
```

rank applies a local ranking operator (3x3 or 5x5), achieving median filtering, erosion or dilation.

```
rank display                         (3x3 median filter)
rank erode 1 to 2 over 5             (shrinks bright regions by about two pixels)
rank dilate; rank erode              (smooths edges)
```

rf applies a two point separable recursive filter to a picture, in all four directions in turn so as to smooth or sharpen with an infinite but symmetric point response. Values of the key a in the range 0 to 1 smooth the image, and values in the range –1 to 0 sharpen it.

```
rf 10 11 a .3
```

sharpen sharpens a picture by adding the result of **hp** to the original (doubling the high spatial frequencies).

```
sharpen display
sharpen 1 over 3                     (milder sharpening)
```

Transformation and filtering

backproject back-projects a 1-D picture into a picture in any direction, that is,'sweeps' it across the picture, adding to or multiplying pixels as it goes.

```
backproject 14 with 10 angle pi/3 multiply
```

ctf (contrast transfer function) generates or applies the transfer functions that describe imaging in high-resolution electron microscopes, including beam tilt, coherence, astigmatism etc. Use the command **help ctf** for a list of variables describing imaging conditions (the command **library em** is a convenient way to initialise them).

```
min=0 max=4 step=.2
ctf squared to display size 128 defocus root(1.5)
ctf 1 to 2 multiply
```

fourier produces the discrete Fourier transform of a picture. Unless the image is complex, only the right half plane of the transform is stored, with the origin at the left.

```
fourier 3 to 4
```

hilbert produces the Hilbert transform, along the rows, of the real part of a picture, storing it in the imaginary part so that the Fourier transform of the result vanishes over the left or right half-plane.

```
hilbert 87 right
```

image regenerates an image from a Fourier or Walsh transform, that is, performs the inverse of the transforms.

```
image 52
image 101 to 102 byte
```

phr randomises the phase of a Fourier transform, preserving the modulus, so that the resulting image can be used to assess any non-randomness in a near-random field.

```
phr 31 32
```

ps produces the power spectrum (the squared modulus of the Fourier transform, or diffraction pattern) of a picture.

```
ps 14
ps 1 14 ln                    (takes the logarithm to compress the range)
```

walsh produces the *Walsh-Hadamard* transform of a picture (a decomposition into irregular square wave rather than sinusoidal components, used for compression and pattern recognition).

```
walsh 1
```

weight multiplies pixels by, or adds to them, a function of radius (their distance from the picture origin) defined in a 1-D picture.

```
weight 10 to 11               (multiplies by function in picture 999 in default)
weight with 13 add            (adds function in picture 13)
```

window filters a Fourier transform so as to pass only pixels near the sites of a lattice defined by variables *u* and *v*, thereby eliminating most of the aperiodic image noise. This is an indirect way of effecting a lattice average.

```
window 8 width .1
```

Line, Lattice and rotational averaging

base adjusts base vectors *u,v* and origin *w* to fit a list of approximate lattice sites stored in a *Plist* class picture, using a least-squares criterion.

```
u=15.6,23.4 v=-20.1,15.2 base 90 verify
base 90 to 91                          (the new list omits off-lattice sites)
```

flc fits lattice Fourier components to local peak profiles in a Fourier transform, either individually or collecting the results for image regeneration. This is an indirect way of effecting a lattice average.

```
u=15.6,23.4 v=-20.1,15.2 flc line 2,3 verify  (fits order 2,3 only)
flc 46 radius 200 to 47; image (fits all peaks within 200 pixels of the origin)
```

lattice marks lattices on a display, and generates lattice site lists in a *Plist* class picture.

```
lattice display:2 spacing 20
u=15.6,23.4 v=-20.1,15.2 lattice size 1024 to 48
```

motif averages a picture over a list of positions taken from a *Plist* class picture.

```
motif 1 to 2 with 90 size 50      (50x50 regions averaged)
motif to 2 size 50 even           (averages even numbered positions only)
```

project produces the 1-D projection or average of a picture in an arbitrary direction.

```
project 10 11 angle .35 average
project 10 horizontally
```

section produces an average radial section over an arbitrary sector of a picture.

```
ps 1 2; section mark display  (right-hand semicircular sector in default)
xwires line; section 20 23 angle theta width pi/10
```

strain deduces the local strain levels for a distorted lattice stored in a *Plist* class picture, and re-sorts the list of lattice points into ascending strain order, with the option of excluding points with a strain level above a given threshold.

```
u=15.6,23.4 v=20.1,15.2 strain 50 to 91
strain 90 to 91 tolerance .1  (omits points with fractional deviation>0.1)
```

Correlation functions and alignment

acf produces the auto-correlation function (*Patterson* function) of a picture.

```
acf 1 to 2
```

ocf produces an orientational cross-correlation function between two pictures rotated about their origins, and sets the variable *theta* to the relative rotation angle at which they match most closely.

```
ocf 29 with 30 radius 10,30
ocf 18 rings 12                    (self-match assesses angular order)
```

rcf produces the cross-correlation coefficient between successive bands of Fourier components from two mutually registered images, thereby assessing the resolution to which they agree. An option provides for phase residual calculation.

```
fourier 1; fourier 2; rcf 1 with 2 to 3; display
rcf phaseresidual 1 with 2 to 3
```

xcf produces the spatial cross-correlation function between two pictures with a common orientation, and sets the variables *x* and *y* to the coordinates of the correlation peak (the relative displacement at which the two pictures match most closely).

```
xcf 10 with 11 to 12; type x,y
xcf 10 with 11 nosearch            (omits the peak search)
```

Particle

analysis

analyse scans through a picture and identifies particles according to given intensity threshold values. Each particle is assigned a unique number or identifier. Various properties, such as size, area, orientation, etc. are calculated for each particle and stored in a *Plist* class picture called the *particle parameter list* (*ppl*). A segmented output picture can also be obtained, with pixel values set to the corresponding particle identifiers (or 0 for the background area). The output from **analyse** can be examined in a number of ways to find out more about the particles found. The picture numbers for the source picture, the *ppl* and the segmented picture are stored in the variables *pimage*, *pplist* and *psegment*. These variables provide the default picture numbers for all of the commands that depend on **analyse**. Type **help particle** for information about all the particle parameters generated by **analyse**.

```
analyse 4 to 5 ge 1.3 le 7.1
analyse 2:12 2:998 ge 1 segment 2:997
analyse 1 area 10 mark display id cm      (omits particles with area<10)
```

pcalculate calculates further parameters for a specified particle, given a source picture and the corresponding *ppl* and segmented picture. The results are listed on the terminal and stored in the variables *si*, *mi*, *xcm*, *ycm*, *mm1*, *mm2* and *phi*.

```
pcalculate id 32 image 1 plist 998 segment 997
```

pcurve calculates the equivalent particle parameters for the area enclosed within the closed curve described by the given *Plist* class picture. If a source picture is also specified, **pcurve** calculates some further parameters (the same as **pcalculate**). The results are listed on the terminal and also returned in variables.

```
xwires curve closed; pcurve
pcurve image 1
```

pdraw allows you to edit a source picture in order to separate particles that **analyse** considers to be merged into single particles. It gets you to input a curve directly on the display and overwrites the corresponding pixels in the source picture with a specified background value. More pixels are reset than with an ordinary curve in order to eliminate any diagonal contact between pixels on either side of the curve.

```
display 1; pdraw image 1 value 0
```

pedit allows you to eliminate particles from a *ppl* or a segmented picture. You specify which particles are to be retained in the output result by means of the keys **if** and **unless**. The keys accept general expressions which may include any of the names that are associated with particle parameters (use the command **help particle** to find out more about these).

```
pedit plist 997 to 996 if area>23 & perimeter^2 <1000
pedit segment 4,5 unless hferet<55      (edits segmented picture 4 into 5)
```

pextract produces an output picture that contains just the source image of a single specified particle (or a simple binary image if no source picture is given) and with all pixels outside the particle set to a suitable background value.

```
pextract id 5 plist 21 segment 22 to 99    (simple binary image)
pextract image 4 to 99 value 2   (particle image output with background = 2)
```

pferet calculates up to 9 feret diameters simultaneously for a given particle. The results are listed on the terminal and are also stored in the variables *f, f2, ..., f9*.

```
pferet id 4 plist 997 segment 996 angle .1,.2,.3,.4,.5
```

phistogram outputs a *Histogram* class picture based on one of the particle parameters, for a selection of particles. The keys **if** and **unless** can be used to specify which particles are to be selected. If the output picture is a display picture, the histogram is drawn on the display.

```
phistogram area to 99 plist 4 if holes   (only for particles with holes)
phistogram angle to display              (for all particles)
```

pid returns the particle id that corresponds to the position given. The id is obtained from the segmented picture. If the point does not lie inside a particle, a zero value is returned. The result is typed on the terminal and stored in the variable *pid*. This variable provides the default value for all of the commands that require a single particle id via the **id** key.

```
xwires; pid segment 6 @xy; ptype all if id=pid
```

pmark marks the selected particles on the display. The annotation can be centred either on the particle reference point or on the centre of area and can be the value of any one of the particle parameters in text form. The keys **if** and **unless** specify which particles are to be marked.

```
pmark                                (mark reference points of all particles)
pmark area if perimeter>100          (mark area if perimeter>100)
pmark cm angle unless holes          (mark cm of particles with no holes)
```

pset stores the specified particle parameters for a selected particle in variables. The keys **If** and **unless** may be used to select a group of particles and the **sort** key, if given, causes these particles to be ordered according to one of the particle parameters. Finally, a single particle is selected from the sorted list by means of the **Index** key (the first particle in default). The option **count** causes the variable *n* to be set to the number of particles first selected. This allows you to scan through the results for a selection of particles and examine the results in more detail.

```
pset area if ~holes sort area descending
pset count if area>25
for index=1,n;pset xcen ycen if area>25; ... ; loop
```

pshow highlights the selected particles on the display. The segmented picture must first have been displayed with unit intensity scaling. The keys **If** and **unless** specify which particles to select and the **sort** key can be used to sort the resulting list according to one of the particle parameters. The selected particles can be highlighted with a range of intensities or colours that vary with the ordering of the particles. The background is displayed in black and non-selected particles are displayed in grey.

```
pshow plist 997 if area>100
pshow if perimeter>120 sort area saturation .8 hue 0,60
```

ptype lists parameter values for a selection of particles or else all of the parameters for a single particle (if you use the option **all**). The keys **If** and **unless** specify which particles to select and the **sort** key can be used to sort the resulting list according to one of the particle parameters. Up to 5 selected parameters can be listed in tabular form. When option **all** is used, the **Index** key selects a single particle from the list (the first particle in default).

```
ptype area perimeter xcen ycen if area>100
ptype hferet vferet circularity sort area if circularity>.8
ptype all if id=4                          (lists all parameters for particle 4)
pset count if area>100
for index=1,n; ptype all if area>100; loop
```

Morphology

dilate adds pixels to a binary image in a number of ways. The number of passes made over the data may be specified as may the combination with neighbours.

```
dilate display
dilate 50 to 51 times 2          (add a two pixel border to all objects)
```

erode provides various morphological manipulations on binary pictures. The **skeleton** key will skeletonise an image. Other options and keys allow you to specify how the erosion takes place and how many passes are made over the source image.

```
erode display
erode 50 to 51 times 3           (erode a three pixel border off all objects)
erode 1 skeletonise              (erode objects down to their skeleton)
erode 5 to 6 outline             (produce outlines of objects)
```

median smooths binary images, removing isolated pixels and lines and similar point and line holes within objects, at the same time.

```
median display
median 50 to 51
```

Remote sensing

box classifies a picture using the *box* or *parallelepiped* method. The **learn** command provides the statistics of each class. This is the fastest of the three classification methods.

```
box 1 to 12 with 3
```

covariance calculates the covariance of a multispectral picture. Additionally the mean, standard deviation and range of each layer is recorded.

```
covariance 1 to 3
```

destripe corrects a source picture for difference in line sensor characteristics. It is possible to adjust by either the mean or the mean and standard deviation (the **mode** option). A reference mean and standard deviation may be specified, if required.

```
destripe 46 to 47 lines 6          (6 line sensors)
```

learn calculates the statistics of training areas in preparation for classification. The covariance, mean, standard deviation, minimum and maximum of each area are calculated. The output picture is multi-layer, with each layer describing one training area.

```
learn 2 to 3 data 9,8,31           (Plists 9, 8 and 31 describe areas)
```

likelihood classifies a picture using the *maximum likelihood* method using the statistics gathered by the **learn** command. Both thresholds and *a priori* probabilities may be given to classes.

```
likelihood 1 to 10 with 3                    (no probabilities or thresholds)
likelihood 23 to 25 with 24 threshold 90     (threshold of 90%)
```

mindistance classifies a picture using the *minimum distance to means* method using the statistics obtained by the **learn** command. A threshold (in terms of standard deviations from the mean) may also be given.

```
mindistance 1 to 11 with 3
```

pct calculates the principal component transform of a multispectral image. The transform may be either forwards or backwards. The output picture may be stretched if required. The eigenvalues and vectors are available for inspection. A further option allows the transform to be of the *Hotelling* variety in which the mean is subtracted, centering the output about the origin.

```
pct 1 to 8 covariance 4 eigen 5
```

rhistogram calculates a 2-D histogram between the bands of a multi-spectral picture. As with the **histogram** command, output may be directed to the display. The range over which the histogram is to be generated is selectable as is the number of channels.

```
rhistogram 46 to 48 min 0,0 max 64,64 preset
```

warp corrects a picture for geometric distortion. The coefficients of an arbitrary order polynomial are calculated using the *least squares* method, with elimination of severe errors if required. The output picture is then generated by re-sampling the source picture using nearest neighbour, bilinear or bicubic resampling.

```
warp 1 to 2 map 3 image 4 verify        (display results of fit)
```

User interface creation

cell controls the behaviour of the cell element. Cells may be positioned on panels or menus. There are four different styles of cell which provide various methods of highlighting.

```
cell id z00 title select   (the contents of the cell are taken from the picture title)
cell create drop add m     (add a cell to menu m)
cell changes 'type ''hit'''   (type "hit" when the cell changes state)
```

device allows a Semper program to determine the limits of the display, redraw the screen completely and also stack up cursor positions.

```
device query 1             (find size of display and number of colours)
device refresh 1           (redraw the display)
```

execute defines what actions will be performed before and after every Semper command sequence.

```
execute on                 (enable execute processing)
execute after 'type sel'   (type current value of select after each command)
```

justification controls how user interface objects are positioned on the display. By default, elements are centred about their specified position.

```
justification top left
justification top
```

menu controls the creation and operation of the menu type element. Three styles of menu exist: *fixed*, *pulldown* and *popup*. Menus can have actions (Semper commands) defined which are executed when they are displayed or hidden.

```
menu id mnu deactivate
mouse query;menu id m position uix,uiy activate   (activates a menu at
                                                   the current mouse position)
```

mouse defines the actions and/or position of the mouse. You can find out the current position of the mouse using the **query** option.

```
mouse id txa                    (move the mouse to the specified element)
mouse query; type nbu           (find how many buttons are on the mouse)
```

panel controls the creation and operation of Semper 6 *Plus* panels. Panels may be fixed (permanently visible) or transient. The **mandatory** option forces interaction with a particular panel.

```
panel create auto transient mandatory
panel id p show
```

textfield controls the creation and operation of Semper 6 *Plus* textfields. These objects can be used to contain Semper commands to be executed. Textfields may only be placed on panels.

```
textfield id t1 contents 'display sel'
textfield id t1 execute
```

uif controls the loading, saving and execution of user interfaces. The **exit** option returns control to the standard command line interface. Other options exist for returning control to the menu interface, displaying the current state of the system, etc.

```
uif enable
uif go
uif status
```

Miscellaneous

event provides command level access to internal Semper event queues.

```
event open pointer          (opens the pointer queue)
event count keyboard        (counts the number of key presses)
```

monitor provides system operation diagnostic information. *This command is provided primarily as a diagnostic tool.*

```
monitor io
monitor rowio on
monitor off
```

null is the command executed by default if no command name is found when interpreting the current command. The ***null*** command itself is normally defined as a synonym for the **display** command.

```
2:3                         (displays picture 2003)
```

pack types packed integer values for each name given.

```
pack size,si2,si3,position,po2
```

unpack types three-character names for each packed integer value given.

```
unpack 30786,30792,30793,26219,26232
```

user is provided as a dummy command with a ready-made Fortran skeleton and interface with the Semper interpreter, which you may adapt as you like while learning how to write your own processing modules for Semper.

```
user 1 to 2 op1 ke1 .2 ke2 pi/3
```

Chapter 3

INSTALLATION SPECIFIC COMMANDS

Overview

This chapter lists the Semper commands that are specific to certain installations, that is, commands that apply only to particular hardware combinations. To find out which of these commands apply to your system, type the following command:

```
help installation
```

or refer to the table given overleaf.

Note that some of the commands listed in this chapter (for example, **assign...display**, **overlay**) have different keys and options depending on the type of installation. To find out the keys and options for a command you can either use the **syntax** command, for example:

```
syntax overlay
```

or refer to the *Command Reference* notes that are supplied with your installation.

The installation-specific commands are grouped according to function. A list of functions headings, in the order that they appear in this chapter, is given below:

- framestore configuration and control
- image acquisition
- file reading/writing
- miscellaneous

If you know the name of a command, but are unsure of which heading it falls under, refer to the *Index*, which lists all commands in alphabetical order.

PC hplj, input, output, spawn	
Data Translation DT2861	assign...display, ilut, live, overlay, slowscan, video
Imaging Technology PCVISIONplus	assign...display, ilut, live, overlay, snap, video
Massram	animate, annotate, grab, live, overlay, screen snapshot
Matrox PIP512/PIP1024	assign...display, ilut, live, overlay, video
Metrabyte MV1	assign...display, ilut, live, snap, overlay, video
MRC500	assign...display, cget, cput, fscontrol, fsxecute, live, overlay, screen
Quantimet 520	assign...display, live, overlay, qget, qput, qshade snap, video
Synapse	assign...display, live, overlay, snap
Synergy	assign...display, fscontrol, fsxecute, live, overlay photo, sscan, ssgen
Imaging Technology FG-100 framegrabber	grab, ilut, video
VAX **overlay, spawn**	
Imaging Technology FG-100-Q	ilut, itech, live, overlay, snap, video
MicroVAX/GPX and VAXstations	overlay, uis
MicroVAX/GPX and VAXstations running DECwindows	overlay, x11
Silicon Graphics	
Imaging Technology FG-100-V framegrabber	assign...display, input, output, overlay
Stanford IRIS 4D installations	sget, sput, ssheet
Sun **assign...display, overlay**	
Hewlett Packard Running X-windows **overlay, x11**	
Lynwood J730 **overlay, snap**	

Framestore

configuration and control

assign...display specifies the individual configuration of a framestore. It changes configuration details such as i/o and memory base addresses, video standard, sampling rate and the number of frames that are present (where the framestore is programmable in these respects).

```
assign display iobase 528 ntsc
assign display membase 12 rate 12.5 frames 1
```

fscontrol is an interactive framestore control command for the *MRC500* and *Synoptics* Synergy processor. It allows you to execute framestore commands, read and write framestore program memory etc.

```
fscontrol
```

fsxecute provides direct control of the *MRC500* or *Synoptics* Synergy framestore from Semper. It allows you to execute framestore commands, reset the board etc.

```
fsxecute reset
fsxecute put 3 cmd 9
```

itech allows you to read or set any of the control registers of the *Imaging Technology FG-100-Q* framestore. It also allows access to any of the framestore's memory locations. *This command is provided primarily as a diagnostic tool.*

```
itech read all
itech reset initialise clear
```

overlay specifies the colour displayed for the overlay plane. Note that some installations have several overlays. In these cases, it is often possible to make the overlays transparent.

```
overlay red
overlay rgb 80, 20,10
overlay cursor on
```

screen allows you to control the way the monitor output is generated.

```
screen off overlay
screen noautomatic
```

ssheet controls the Semper 6 *Plus* interactive 3D surface viewer.

```
sssheet 2:1
```
(displays the surface described by the intensity values in picture 2:1)

uis controls the refresh rate of the front display of a *DEC VAXstation*.

```
uis refresh
```

x11 controls the rate at which the display is updated.

```
x11 ilimit 4
```
(refreshes the display after every 4 image writes)

Image

acquisition

animate allows you to view sequences of frames at video rate.

```
animate display:1
animate display:2 continuous
animate display:1 layer 7,1 dwell 2
```

grab allows you to grab images or a sequence of images from the video input.

```
grab
grab 2:3 wait 0
```

live makes Semper grab frames continuously from a video-rate source for a fixed or variable period. You can specify a time period for video input and, depending on your installation, a channel number, recursive filter, gain and offset for the video input signal and destination partition number etc.

```
live wait 5
live partition 2 channel 1
live time 1 gain 5 offset 2
```

slowscan makes Semper grab frames continuously from a slowscan source for a fixed or variable period,

```
slowscan
slowscan wait 5
slowscan frame 3
```

snap takes a snapshot of a single frame from a video-rate source. Depending on your installation you can specify an input look-up table, a channel, video input signal gain and offset, destination partition etc.

```
snap channel 2 partition 1 preset
snap gain 7 offset 5 partition 1
```

snapshot allows you to grab images or sequences of images from the video input.

```
partition 1 frame 1; snapshot to display:1
```
(grabs a single image into frame 1)

sscan grabs frames, or part of a frame, continuously from a slowscan source. You can grab a frame into the whole screen or a partition of the screen. You can also apply noise filtering (recursive, averaging or *Kalman*) to the input.

```
sscan
sscan channel 2 partition 2 frames 25 time 3
sscan movie partition 3 dwell 3 kalman frames 15 ignore 100,200
```

video sets up a framestore in preparation for grabbing frames from an image source. It can define the image capture window limits, the offset and gain to be applied to the input signal, the input look-up table etc., depending on your type of installation.

```
video reset
video start 0,0 end 255, 511
video camera 2 offset 5.8 gain 0.9 pll
```

File

reading/writing

cget reads a DOS file which contains an image written by the *MRC500 microscope control software.*

```
cget 23 name 'two'
```

cput writes a picture to a DOS file that can be subsequently read by the *MRC500 microscope control software.*

```
cput 23 name 'one'
```

input reads pictures from files created by the Semper 6 **output** command, from dump files created by MicroSemper1, or from other selected programs, such as *Media Cybernetics* CUT file format.

```
input 5 name 'newdata'
input 20 dump name 'olddata'
```

nget reads *Sony .vid* format files into layers of existing Semper pictures.

```
nget to 5 name 'newdata.vid' layer 2
```

nput outputs layers of a Semper picture in *Sony .vid* format.

```
nput from 5 name 'data' layer 4
```

output writes pictures in a fast binary form that can be read by the Semper **input** command or in a format that can be read by MicroSemper 1 and Figment 1 systems. It also writes images in *Media Cybernetics* CUT file format.

```
output 23 name 'binary' unlabelled
output 11 new cut name 'export'
```

qget read a *Quantimet* 520 format image file into a Semper picture.

```
qget name '520PIC' to 2
```

qput writes a Semper picture to a disc file in *Quantimet* 520 format.

```
qput name '520PIC' from 3
```

sget reads an image from a *Stanford* format file image.

```
sget 4 name 'stanford.pic'
```

sput writes a picture to a *Stanford* format image file.

```
sput 4 name 'stanford.pic'
sput 4 name 'stanford.pic' new
```

Miscellaneous

annotate

hplj prints pictures on a *Hewlett-Packard LaserJet+* (or equivalent printer) connected to an IBM PC or compatible, using the parallel port. **hplj** also prints display partitions complete with their overlays.

```
hplj 21
hplj partition fs:4
```

ilut manipulates the input look-up table. It allows you to copy an input lut, read back luts from the framestore and store luts as Semper pictures.

```
ilut reset
ilut like 1 to 4:25
```

photo outputs the contents of the display (or display partition) to a recording CRT, such as those found in scanning electron microscopes. (Use **photo** if your Synergy framestore has a *Photo Output* facility).

```
photo
photo partition 2 frames 10
```

qshade controls the *Quantimet* 520 shading corrector to reduce the effect of background image distortions.

```
qshade set
qshade on
```

spawn escapes temporarily from Semper to the local operating system.

```
spawn
```

ssgen controls the scan waveforms and pixel clocks if your Synergy framestore has a Scan Generator.

```
ssgen
ssgen xactive 100 yactive 200
```

ssheet controls the Semper 6 *Plus* interactive 3D surface viewer.

```
ssheet 2
ssheet from 4 to 5
ssheet 9 colour 2 emission 5
```

Index

Symbols

@name, 2-6

@number, 2-6

A

acf, 2-29

add, 2-6

analyse, 2-30

animate, 3-1

ask, 2-3

assign, 2-17

assign...display, 3-3

B

backproject, 2-25

base, 2-27

beep, 2-3

bell, 2-3

box, 2-34

break, 2-6

buzzer, 2-3

C

calculate, 2-19

cell, 2-36

cget, 3-7

close, 2-17

cls, 2-3

command,
- by function, 2-2
- keys, 1-2, 3-1
- installation specific, 1-1, 3-1 to 3-10
- options, 1-2, 3-1
- syntax, 1-2, 1-3

compress, 2-17

continuation line, 1-2

contour, 2-10

copy, 2-14

correct, 2-19

correlation functions
- and alignment, 2-29 to 2-30

covariance, 2-34

cput, 3-7

create, 2-14

ctf, 2-25

cut, 2-21

D

deassign, 2-17

delete, 2-14

destripe, 2-34

device, 2-36

device management, 2-17 to 2-19

differentiate, 2-23

dilate, 2-33

directory, 2-17
display, 2-10
documentation, Semper, 1-1

E

echo, 2-3
edge, 2-23
edit, 2-6
end, 2-7
erase, 2-18
erode, 2-33
event, 2-38
examine, 2-10
execute, 2-36
exit, 2-7
extract, 2-21

F

file reading/writing, 3-7
find, 2-21
fir, 2-23
fit, 2-19
flc, 2-27
flush, 2-18
for, 2-7
fourier, 2-25
framestore configuration and control, 3-3
fscontrol, 3-3
fsxecute, 3-3
fullplane, 2-21
function, Semper commands, 2-1

G

gaussian, 2-19
geometric operations, 2-21 to 2-22

H

halfplane, 2-21
help, 1-2, 2-3
    installation, 3-1
hilbert, 2-25
histogram, 2-10
hp, 2-23
hplj, 3-9

I

if, unless, 2-7
ilut, 3-9
image, 2-25
image acquisition, 1-1, 3-5 to 3-6
inkey, 2-4
input, 3-7
installation specific
    commands, 1-1, 3-1 to 3-10
itech, 3-3

J

jump, 2-7
justification, 2-36

K

keys, 1-2, 3-1

L

ladjust, 2-11
lattice, 2-27
learn, 2-34
letter, 2-14
library, 2-7

likelihood, 2-34
line, lattice and rotational averaging, 2-27 to 2-28
list, 2-7
live, 3-5
lmean, 2-23
local, 2-4
local operations, 2-23 to 2-24
log, 2-4
loop, 2-8
lorentzian, 2-19
lsd, 2-24
lset, 2-11
lut, 2-11
lvariance, 2-24

M

macro, 2-8
magnify, 2-22
manuals, 1-1
map, 2-20
mark, 2-11
mask, 2-20
median, 2-33
menu, 2-36
mindistance, 2-34
miscellaneous commands, 2-38
installation specific, 3-9 to 3-10
monitor, 2-36
morphology, 2-33 to 2-34
motif, 2-27
mouse, 2-37

N

name=val, 2-3
negate, 2-20
next, 2-8
nget, 3-7
noise, 2-20
nput, 3-7
null, 2-38

O

ocf, 2-29
on-line help, 1-2
options, 1-2, 3-1
order, 2-8
origin, 2-14
output, 3-7
overlay, 3-3

P

p, 2-4
pack, 2-38
page, 2-4
panel, 2-37
particle analysis, 2-30 to 2-32
partition, 2-18
paste, 2-22
pcalculate, 2-30
pcb, 2-15
pct, 2-35
pcurve, 2-30
pdraw, 2-30
peaks, 2-22
pedit, 2-31
pextract, 2-31
pferet, 2-31
phistogram, 2-31
photo, 3-9

phr, 2-25
picture display and inspection, 2-10 to 2-13
picture management, 2-14 to 2-16
pid, 2-31
pixel, 2-4
pmark, 2-31
pointer, 2-4
postscript, 2-11
print, 2-12
program execution, 2-6 to 2-9
project, 2-27
ps, 2-26
pset, 2-32
pshow, 2-32
ptype, 2-32

Q

qget, 3-7
qput, 3-8
qshade, 3-9
quit, 2-8

R

ramps, 2-18
rank, 2-24
rcf, 2-29
read, 2-15
reclass, 2-15
reinitialise, 2-18
remote sensing, 2-34 to 2-35
rename, 2-8
renumber, 2-15
report, 2-4
return, 2-8
rewind, 2-18
rf, 2-24
rhistogram, 2-35
rotate, 2-22
run, 2-8

S

Semper, manuals, 1-1
Semper 6 Command Reference, 1-1
Semper 6 Command Summary, 1-1
Semper commands, by function, 2-1, 3-1
save, 2-15
scale, 2-20
screen, 3-3
section, 2-27
select, 2-15
separate, 2-22
sget, 3-8
sharpen, 2-24
sheet, 2-12
show, 2-5
slowscan, 3-5
snap, 3-5
solid, 2-12
spawn, 3-9
spc, 2-12
sput, 3-8
sscan, 3-6
ssgen, 3-9
stack, 2-22
stop, 2-9
strain, 2-28
survey, 2-12
syntax, 1-2, 2-5, 3-1
syntax, Semper commands, 1-2 to 1-3

T

textfield, 2-37
time, 2-5
title, 2-15
transpose, 2-22
turn, 2-22
type, 2-5

U

uif, 2-37
uis, 3-4
unpack, 2-38
unset, 2-5
user, 2-38
user interface commands, 2-36 to 2-37

V

variables and terminal information, 2-3 to 2-5
video, 3-6
view, 2-12

W

wait, 2-9
walsh, 2-26
warp, 2-35
weight, 2-26
xwires, 2-13
window, 2-26
wp, 2-16
write, 2-16

X

xcf, 2-29

Y

ymod, 2-13