Introduction to the use of *TIMWIN*

This document offers a very compact description of various aspects of ***TIMWIN*** usage.
TIMWIN is a program for image processing and -measuring, and runs on an IBM-PC/AT or any other MS-DOS compatible computer under Microsoft Windows. A frame grabber can be used for image acquisition and display, but image display can also take place in the Windows environment.

Image processing commands can be issued in two ways:

- using menus and dialog boxes
- using the command line

Dialog boxes guide the user through the various options, so they offer a secure way to enter a command. However, this method is slow and requires many actions, so after a short introduction you may prefer the command line method.
If a command is specified through a dialog box action, the resulting command string is printed in the system messages area and the command history as well.


fig. 1 Main ***TIMWIN*** window

Commands

In ***TIMWIN***, about 190 image processing commands are available. They are divided in groups, called families. Below the families are introduced.

Pixel
Operations, that operate on the image pixel by pixel. Functions: arithmetic, logic, table look-up. Monadic pixel operations operate on one source image, diadic pixel operations operate on two source images.

Window
Linear and non-linear neighbourhood operations, that use the pixels in a region around a pixel to determine the result. Also called: filters and neighbourhood operations

Bitplane
Manipulation of bitplanes: combining, exchanging, moving.

Cellular Logic
Binary morphological operations: erosion, dilation, contour, skeleton, etc.

Geometric
Operations that influence the position of the pixels: rotation, scaling, moving , etc.

Miscellaneous
Generation of test images, image editing, erasing, etc.

Parameter
Operations whose purpose it is to produce data (single values or arrays). Generally an image

is produced as well.

Transport
Copying between images, or images and disk files.

Graphic
Drawing lines, circles, plotting graphs.

Control
Controlling the program and frame grabber functions.

Fourier transforms
Transformation to and from, and filtering in the Fourier domain

If you look for a particular command and you don't know the name, you will find it easily by first determining its family. Then you can use the following methods to get the particular command:
- In the ImageProc menu, select the family name and make your choice from the commands that show up in the sub menu.
- Activate the Help function by pressing the [F1] key or clicking a Help button. Then click the Contents button; in the next window click Commands and families. Then click the family name you're interested in.

For example, you want to perform a threshold operation to make a grey value image binary. Since this is a pixel operation, you select Pixel (monadic) in the ImageProc menu.

Specifying a command

The general format of a command (as entered on the command line) is:

<command> <image spec> <numerical specs> <special specs>

Example: `thre a 100 /`

Here **thre** is the command, **a** is the image, **100** is a numerical parameter and **/** is the special parameter.
The on line help explains the parameters and the syntax.

Images

TIMWIN operates on images, that must be present in either computer or frame grabber memory, or as a disk file.
Images can be visible (if they are located in a frame grabber or if a display window is attached) or invisible (if they are located in computer memory or on disk).
In ***TIMWIN*** the image structure is user definable, but in this document we assume a standard environment, as determined by the standard **images.tim** file.
Image size is 256x256 or 512x512 pixels, while each pixel is a byte (8 bits). These images are addressed by a single letter.
The status bar (see fig. 1) shows the available images.

TIMWIN Introduction

Image display on a frame grabber equipped system

The following images are available (small differences may result from frame grabber properties):

- In computer memory (invisible): **a, b, c, d**
- In the frame grabber: **p, q, r, s, t, u, v** and **w**
- In a Windows display: **h** and **i**

The images appear on the frame grabber screen as follows:

fig. 2 Images in the frame grabber display

You can use the frame grabber's zoom facility to zoom in on a single image. To zoom in:
- Use the zoom command
- Or click the zoom button in the status window

Image display without frame grabber

The following images are available:

- In computer memory (invisible): **a, b, c, d p, q, r** and **s**
- In a Windows display: **h** and **i**

The image windows can be made visible using the View menu, or by using the **show** command.

Active image

The active image is the image in which the image processing results appear. This will usually be (but is not required) a visible image. The active image is shown in the status bar (see fig. 1), where its button has a red border.

You can select the active image by clicking its button, or by command (**dest**).

Sub images

Sub images are a part of standard images. They are specified by appending 1, 2, 3, 4, or c to the image name: **a1, b2, pc**, etc. The following figure shows the properties of sub images:

fig. 3. Sub images

The fixed sub images occupy ¼ of the image; the variable sub image occupies a variable area at a variable position. The **frmt** and **curs** commands control the sub image properties. The status bar (see fig. 1) shows the sub image properties of the active image.
Some operations deal with sub images (**mark**).

Image specification

In a command, the image parameter specifies the source image of the operation. The result is written to the active image (see page 5), or the default destination). If no source image is specified, the active image is the source as well. See the following example:

```
dis p
add a 10
```

The pixels of image **a** are incremented by 10. The resulting image is written to the destination image (which is **p** in this case).

```
add 10
```

the pixels of the default source are incremented by 10, the result is put back in the same place.

There are a few exceptions to the above rule. Operations, that only modify selected pixels or selected bitplanes, or that produce test images, function in the indicated image. Examples: graphic, bitplane, cellular logic operations.

Parameters

Most commands need further specification. This is done by value(s) (using numerical parameters) or by special characters or strings. The on line help informs you which options can be specified.
Not all parameters have to be specified. Generally, ***TIMWIN*** assumes a default values if a parameter specification is omitted. The documentation informs you about the default values for commands. The default image is the active image (see page 5)
Aliases allow the user to make the commands more legible. Example: bitplanes can be addressed by their number (1, 2, etc.) but also by their colour (as they appear on the screen: red, green, blue).

Bitplanes

In a 'normal' (grey value) image each pixel is defined as a byte. Since a byte consists of 8 bits, this allows us to show each pixel as one grey value out of 256. However, we can also use an 8-bits image to store eight 1-bit images. A one-bit image is useful for storing and processing binary image data. Bitplane- and Cellular Logic operations operate on 1-bit (bitplane) images (actually: a single bitplane in an 8-bits image) In ***TIMWIN***, there is no such thing as a binary image type.

fig. 4 Bitplanes

A bitplane is a horizontal cross-section of an image, capable of storing one binary image. Using special display options we can show any of the bitplanes of an image, thus allowing observation of binary image operations. While processing images you will frequently switch between the grey value- and bitplane representation of images.
The colours, mentioned in the figure, indicate the colours that appear on the screen when the

bitplane display option is selected. See Look Up Tables (page 7)

Return Parameters

Many ***TIMWIN*** operations return a special value, which gives information about the operation and the image being processed. This value is printed in the system area of the main window and can be used as a value in a command file (see page 9). Example:

Command	**Printed message**
`add p 10`	`Number of overflow pixels: 1234`

This message indicates that adding 10 to the pixel values of image **p** results overflow (result >255) for 1234 pixels

Help

TIMWIN has context sensitive help. It is based on Windows' powerful help utility, and allows you to access the information in many ways.
Most dialog boxes have a help button, that will show help information on the concerning item. If you are in the process of entering a command, pressing the help key [F1] brings up help on that command.
Once you are in the Help application, you can browse through the information in many ways.

Look Up Tables

In frame grabbers, look up tables (LUTs) perform a real time pixel transformation of the output image (output LUTs) or the input image (input LUTs). Since there are three output LUTs (one for each of the monitor inputs Red, Green and Blue), that can be individually programmed, a virtually unlimited selection of output transformations is possible.
LUTs are used to display the image appropriately in a given situation. Although Windows images don't have hardware LUTs as frame grabbers do, you can choose the same display options as frame grabbers.
The **lut** command allows you to create an unlimited amount of colour transformations. However, for normal use the following standard LUT patterns are available:

LUT	**Purpose**
1	is used for display of black and white images
2	shows an image in pseudo colours, which magnifies small differences in contrast
3	is as LUT 1, but the content of the lowest (least significant) bitplane is shown in red. This is used for displaying graphics and binary

information together with grey values.

4 is as LUT 3, but the content of bitplane no. 2 is shown in green, and that of bitplane no. 3 is shown in blue. This makes it possible to observe three binary images at a time (in red, green and blue).

A table can be selected as follows:

- By clicking the appropriate arrows in the status bar (see fig. 1)
- By command:

`lut 2 #` (where # is a number between 1 and 4) will select the LUT function # for the frame grabber output LUT

`*winlut #` will load the corresponding display function for Windows images.

Useful features

With each image processing operation you can get additional facilities by enabling special functions. In the Options menu of these windows, you can select the conditions to update them.

Image statistics

To get a number of grey value statistics after each operation, open the statistics window:

- In the View menu, click Statistics Window

Histograms

To get a graphic representation of the grey value histogram, open the graphic window:

- In the View menu, click Graph Window
- In the Options menu of the Graphic window, check Histogram Update.

Frame Grabbers

TIMWIN runs on MS-DOS PC/AT machines, but for acquisition and display of images frame grabbers are used. Various types of frame grabbers are applied, with slightly varying properties. Below a description of the frame grabbers is given. You can use the ver command to find out which frame grabber is in your system.

PCVISION

The PCVision is a simple frame grabber, with one 512x512 or four 256x256 images. It has no provisions for zooming in. There are 4 Look Up Tables for display.

PCVisionPlus

TIMWIN Introduction

The PCVisionPlus has two 512x512 pages, and each of them can be selected for display. 4 images of 256x256 are displayed at a time. You can zoom in, so that one 256x256 image fills the entire display screen. It has 8 Look Up Tables for display.

VFG

This grabber consists of a 1024x1024 memory, in which four 512x512 or sixteen 256x256 images can be stored. A pixel can be 12 bits. It has sixteen Look Up Tables for display and four levels of zoom (x1, x2, x4 and x8). In addition to this the VFG supports real time image processing.
VFG frame grabbers have square pixels, so that the display window is 768x512 and six 256x256 images can be displayed at a time.
A non standard camera (e.g. 1000x1000 Videk Megaplus) can be connected.

Cortex-I

This is a simple frame grabber, with one 512x512 or four 256x256 images. It has no provisions for zooming in. There are no Look Up Tables for display.

Command files

(Note: the command file compiler is not available in the Demo version of ***TIMWIN***).
Command files are programs, consisting of image processing commands and other commands from ***TIMWIN***'s programming language. These special CFI commands include:

- definition and manipulation of variables (numeric, strings), including assigning ***TIMWIN*** return parameters to variables.
- program control using `if - elseif - else, for - next,` etc. constructions, subroutines, nesting of command files.
- various control functions, binary and formatted file IO, user interaction, etc.
- debugging.

To start writing a command source file, open the **CommFile** dialog box, enter a file name and click Edit. The source file will have the extension .cmd.
For details on command file functions, keywords and syntax descriptions, see the on line help: click the Contents button and then Command Files.
Before a command file can be executed, it must be compiled. Compiling takes place automatically, if the compiler switches are set accordingly.. The result of the compilation process is written in the system area of the main window (see fig. 1, page 1). You can also invoke the compiling process from the editor, by clicking **Compile** in the main menu.
To run the command file, click **Run** in the edit menu. You can also use the command line: `/commfile` or `*commfile` will start the command file `commfile.cmc`. To run a command file in debug (single step) mode, append `debug` after the command. Example: `/commfile debug.`

Example

The following is an example command file, that shows some of the items in a command file.

```
;course -- demonstration program for
;          TIMWIN command files
;purpose: measure the size of objects in an image
;expects: image to be processed in 'a' (e.g.: "cermet")
;****************************************
int number             ;define variables
int pass
int area
              ;Start of program
thre a -128            ;threshold image a with 128, invert result
number = label         ;label objects; find number of objs
for pass = 1 to number step 1 ;beginning of loop
 area = mark pass     ;get next individual object, store area
                      ;print measurements
 print "Object ", pass, "has area: ", area
next                   ;loop until done
stop                   ;ready
```

This simple command file is a good start for one of the exercises (measuring objects). You are advised to experiment with this example (run it, modify it) in order to get acquainted with the command file procedures.

Guided Tour through *TIMWIN*

The following paragraphs contain a guided tour along several important ***TIMWIN*** areas.
It is advised to look up the description of the commands when you execute them, to find out about options, etc.
There are small differences for systems with a frame grabber and systems without one. They occur mainly in image specifications. If such differences arise, they are indicated in the middle (WinIm) column. Replace the image specification in the command with the specification in the WinIm column.
(Frame grabber only:) when you start this tour, the system is supposed to be initialised. To be sure about that, you can run command file **init** (command: `/init` or `*init`).
(Windows images only:) To make the images visible, check them in the **View** menu or use the **show** command.
The images used during the tour have size 256x256.

Display and other images

The following shows how to control the active image in display.

Command (FRGrab)	WinIm	Description
`dis p`	`h`	Select an image for destination of operations and display

`wig`		Write a wedge of grey values there
`dis q`	i	Select another image for destination and display
`dis miss`		Copy image "miss" from disk into q (i). The **dis** command is used to copy images *OR* to change fr.grabber display, depending on the argument.
`copy cermet r`	`h`	Another way of copying images: specify destination as well.
`zoom`	(n.a.)	The 256^2 image now fills the entire screen (useful to observe details. Depending on the frame grabber type 0, 1 or 3 levels of zoom are available.).
`zoom 0`	(n.a.)	
`zoom 1`		Zooming can also be controlled directly: specifying the level of zooming.
`dis s`	`i`	Make **s** (**i**) the active image
`inv p`	`h`	Image **p** (**h**) is used as a source to be inverted; the resulting image is placed in **s** (if display is zoomed in you'd better zoom out to be able to observe both images)
`inv`		Since no image is specified, the active image (**s/h**) is used as a source
`save a`		The active image is saved in image **a**. **a, b ,c** and **d** are additional images, that can be used to store intermediate results. They are located in computer memory, so their content is not visible. To get an image back, use the **copy** or **dis** command

Sub images

This section shows management of sub-images.

Command (FRGrab)	**WinIm**	**Description**
`dis p`	`h`	Select an image
`inv p1`	`h1`	The upper left quarter of the image is inverted. Appending 1, 2, 3 or 4 to the image name addresses a quadrant of the image.
[F5]		This is the cursor control key. Repeat pressing this key until a rectangle appears (notice the varying cursor shapes).
`inv pc`	`hc`	The part of the image inside the rectangle is processed. Appending a 'c' to the image name addresses a sub image, which is determined by the cursor position and sub image format.

`frmt 88 55`		Change the sub image size by command (observe the status bar). You can also use the arrow buttons in the status bar for this purpose
`curs 111 177`		Change the cursor position by command. You can also use the arrow buttons in the status bar.
`inv pc`	`hc`	Another part of the image is processed.
`dis q`	i	Change the active image
[F5](3x)		Hit this key until a rectangle appears
`inv pc`		The processed image part is determined by the cursor position of the source image; the position of the result is determined by the cursor position of the active image. These two can be locked (see the description of the curlock command)

Grey value operations

This section shows a few grey value operations. Feel free to change the destination image whenever you want to compare results with previous operations.

Command (FRGrab)	**WinIm**	**Description**
`dis p`	`h`	
`dis miss >a`		Get an image and store it into a for later use (> in front of an image means: post transport). Note that the source image remains unchanged.
`add a 20`		Pixel operation: increment each pixel with 20
`cstr a`		Pixel (table) operation: Stretch contrast
`comp a > 128`		Parameter operation: compare pixels. Note return value
`grad a` `cstr`		Neighbourhood operation: gradient (1st derivative). The contrast stretch makes the result more clear.
`lapl a` `cstr`		Neighbourhood operation: Laplace filter (2nd derivative)
`robg a` `cstr`		Neighbourhood operation: edge detection
`shrp a`		Neighbourhood operation: sharpen image (=laplace+orig)

`unif a 11` Neighbourhood operation: uniform blur (window size: 11x11)

`kuwa a 9` Non linear neighbourhood operation: uniform filter while preserving edges.

Bitplanes

Understanding the concept of bitplanes is very important. Take good notice of the examples presented here. If there is anything in the examples that you don't understand, study the help text on this subject.

Command (FRGrab)	WinIm	Description
`dis p`	`h`	
`zoom 1`	(n.a.)	Make p the active image and zoom in
`dump p1 0`	`h1, h2`, etc.	
`dump p2 50`		
`dump p3 100`		
`dump p4 200`		
`dump pc 255`		First some grey values. The **dump** command fills the image with a single value. Normal display shows low pixel values dark, and high values bright. Since each pixel consists of 8 bits, 2^8 (=256) values of grey can be displayed.
`era`		Erase the image
`lut 2 4`	`lut 3 4`	This command switches display to bitplane mode. A bitplane is an image of which the pixels occupy only a single bit. The 8 bits in a grey value image can be used to show 8 independent binary images: one in every bitplane. To reflect this, display mode must be changed. The lowest 3 bitplanes (1, 2 and 3) now display red, green and blue respectively.
`bdump p1 red`	`h1`	Fill bitplane 1 partially. red is an alias for 1. You may use red, green and blue for 1, 2 and 3 to indicate bitplanes.
`bdump pc green`	`hc`	
`bdump p4 blue`	`h4`	Idem bitplanes 2 and 3. The image now contains three individual binary images. Notice that (on display) green overlays red, and blue overlays both the others.
`lcon blue`		
`lsk green`		
`ler red 10`		
`bor red green blue`		You can use bitplane commands (starting with a *b*) and Cellular Logic commands (starting with an *L*) to operate on bitplanes. Using commands, meant for grey value operations, with images containing bitplanes is not

prohibited, but may give unexpected results. Always realise the type of operations and images that you are working with.

Measuring

Several ***TIMWIN*** commands return values. These values appear in the system message area (see fig. 1, page 1).

Command (FRGrab)	WinIm	Description

`dis p`
`lut 1`
`lut 3 1`
	`h`	Set display in grey value mode

`dis cermet`		Get an image consisting of several objects.
`thre`		Threshold the image (make it binary). All 8 bitplanes now contain he same information. The return parameter indicates that thre found 131 to be a useful threshold value.
`inv`		Invert the image to make the object pixels to be set and the background pixels to be reset.
`label`		This operation assigns the pixels, belonging to a single object, a unique grey value. The return parameter shows, that label found 66 objects. Since grey values 1 to 66 are assigned, the objects may appear dark.
`lut 2`	`/winlut 2`	Use another look up table to enhance contrast. This one assigns pseudo colours to the grey values.
`comp p > 0`	`h`	This operation counts all pixels that match the specified relation, and change their value to 255 in addition. The return parameter shows that the number of object pixels is 22434. Thus the mean number of pixels (= area) is 22434/66 = 339.
`label`		Label the image once more
`mark 11`		**mark** searches the image to find the 11th object. It returns the number of pixels of the object. In addition it adjusts the cursor position and the sub image format in accordance with the object, as can be seen in the status bar. See the example command file at page 9 for a method to process all objects automatically.