berode

keys:	**[from]**	*<number>*	source picture
	[to]	*<number>*	destination picture
	with	*<number>*	picture specifying structuring element data
	mask	*<number>*	mask picture to restrict processing to specified region
	times	*<number>*	number of times to apply structuring element data
	edge	*<number>*	edge value
options:	**source/output**		fix value of source/output edge pixels according to the value of the **edge** key

You use **berode** to carry out generalised binary erosion of foreground regions. The way in which the erosion is carried out is determined by the structuring element specified by means of the **with** key. The result of applying each structuring element is obtained by positioning the origin of the structuring element over each source pixel and outputting a 1 if the structuring element matches the source image in that position. The technique is most often used to remove layers of pixels round the boundaries of foreground regions, which is why it is called **erosion**.

Examples

```
create 99 size 3 value 1; berode 1 with 99 times 1
```

This example removes one layer of pixels from all foreground objects in picture 1 (same result as **erode 1 times 1**)

```
create 99 size 3 value 1; berode 1 with 99 times 4
create 99 size 9 value 1; berode 1 with 99 times 1
create 99 size 3,3,4 value 1; berode 1 with 99 times 1
```

Each of the above examples removes four layers of pixels from all foreground objects in picture 1 (same result as **erode 1 times 4**)

```
create 99 size 9,9,2 value 0
for i=-4,4; pixel i,0,-1=1; pixel 0,i,0=1; loop
berode 1 with 99 times 1
```

This example erodes foreground objects in picture 1 with 9 by 9 square structuring element which has been decomposed into two much reduced structuring elements (gives same result as previous example but in less time).

berode

Description

The source picture is treated as a binary image: zero values denote background pixels and non-zero values denote foreground pixels.

Each layer of the picture specified by means of the **with** key defines a structuring element to be applied to the source picture. If there is more than one layer, the structuring elements thus defined are applied in sequence, starting with layer 1. Non-zero pixels define the components of a structuring element. Any zero pixels are ignored. The origin of the picture defines the origin of the structuring element.

For example,

```
0 0 0 0 0                                0 0 1 0 0          1
0 0 1 0 0          1                     0 0 1 0 0          1
0 1[2]1 0    =>  1[1]1                   0 0[1]0 0    =>  [1]
0 0 1 0 0          1                     0 0 1 0 0          1
0 0 0 0 0                                0 0 1 0 0          1
```

where [ ] marks the origin of the picture and the structuring element.

The process of eroding an object with a structuring element can be explained in simple terms if you think of the structuring element as a binary template. The origin of the structuring element is positioned over each source pixel in turn and a 1 is output if the structuring element matches the source image, that is, if every non-zero component of the structuring element coincides with a non-zero pixel in the source image. Otherwise a 0 is output. Erosion functions just like a Hit or Miss transform where the structuring element defines only *hits*. For more details about Hit or Miss transforms consult the documentation for the **bhmt** command.

For example,

```
0 0 0 1 0                                0 0 0 0 0
0 1 1 0 0                                0 0 1 0 0
0 0[1]1 1        (-)  1[1]       =       0 0[0]1 1
0 1 1 0 0                                0 0 1 0 0
0 1 0 0 0                                0 0 0 0 0
```

where (–) denotes erosion.

An equivalent definition for erosion can be obtained by combining translates of the source image.

The result is the intersection (logical AND) of the translates of the source image where the position of the origin of the structuring element with respect to each non-zero pixel in the structuring element defines an offset to be applied to the source image.

berode

For example

```
0 0 0 0 0          0 0 0 0 0          0 0 0 0 0    0 0 0 0 0
0 0 1 0 0            0 1              0 0 1 0 0    0 0 0 0 0
0 1[1]1 0   (-)  [1]0     =   0 1[1]1 0   (&)    0 1[0]0 0  =  0 1[0]0 0
0 0 1 0 0                    0 0 1 0 0          1 1 1 0 0    0 0 1 0 0
0 0 0 0 0                    0 0 0 0 0          0 1 0 0 0    0 0 0 0 0
```

where (&) denotes intersection.

With a multi-layer picture you can specify a series of structuring elements to be applied in turn. For example, the following result

```
0 0 0 0 0 0 0                          0 0 0 0 0 0 0
0 0 0 1 0 0 0                          0 0 0 0 0 0 0
0 0 1 1 1 0 0          1 1 1           0 0 0 0 0 0 0
0 1 1[1]1 1 0    (-)  1[1]1    =       0 0 0[1]0 0 0
0 1 1 1 1 0 0          1 1 1           0 0 1 1 0 0 0
0 1 1 1 1 0 0                          0 0 0 0 0 0 0
0 0 0 0 0 0 0                          0 0 0 0 0 0 0
```

can be obtained with fewer operations by applying two simpler structuring elements

```
0 0 0 0 0 0 0                          0 0 0 0 0 0 0
0 0 0 1 0 0 0                          0 0 0 0 0 0 0
0 0 1 1 1 0 0          0 0 0           0 0 0 1 0 0 0
0 1 1[1]1 1 0    (-)  1[1]1    =       0 0 1[1]1 0 0
0 1 1 1 1 0 0          0 0 0           0 0 1 1 0 0 0
0 1 1 1 1 0 0                          0 0 1 1 0 0 0
0 0 0 0 0 0 0                          0 0 0 0 0 0 0

0 0 0 0 0 0 0                          0 0 0 0 0 0 0
0 0 0 0 0 0 0                          0 0 0 0 0 0 0
0 0 0 1 0 0 0          0 1 0           0 0 0 0 0 0 0
0 0 1[1]1 0 0    (-)  0[1]0    =       0 0 0[1]0 0 0
0 0 1 1 0 0 0          0 1 0           0 0 1 1 0 0 0
0 0 1 1 0 0 0                          0 0 0 0 0 0 0
0 0 0 0 0 0 0                          0 0 0 0 0 0 0
```

berode

Decomposing large structuring elements in this way can significantly reduce the processing time. The structuring element can be decomposed if a set of structuring elements can be found which result in the original structuring element when combined by dilation. The above example illustrates this.

```
0 0 0        0 1 0        1 1 1
1[1]1  (+)   0[1]0   =    1[1]1
0 0 0        0 1 0        1 1 1
```

Where (+) denotes dilation.

For a precise definition of dilation and more information about decomposing structuring elements, consult the documentation for the **bdilate** command.

The whole process can be repeated by specifying the number of iterations with the **times** key. The default is to carry out one iteration. Processing is stopped if no change is detected after applying one complete sequence of structuring elements.

The **mask** key can be used to restrict processing to specified regions of the image, that is, wherever mask pixels are zero, the source pixel value is output and elsewhere, the eroded result is substituted. Of course, the **mask** picture must have the same dimensions as the source picture. For example, the **mask** image

```
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 1 1 1 1 1 1
0 1 1 1 1 1 1
0 1 1 1 1 1 1
0 0 0 0 0 0 0
```

when applied to one of the previous examples

```
0 0 0 0 0 0 0                                0 0 0 0 0 0 0
0 0 0 1 0 0 0                                0 0 0 0 0 0 0
0 0 1 1 1 0 0          1 1 1                 0 0 0 0 0 0 0
0 1 1[1]1 1 0    (-)   1[1]1    =            0 0 0[1]0 0 0
0 1 1 1 1 0 0          1 1 1                 0 0 1 1 0 0 0
0 1 1 1 1 0 0                                0 0 0 0 0 0 0
0 0 0 0 0 0 0                                0 0 0 0 0 0 0
```

berode

gives the following result

```
0 0 0 0 0 0 0        . . . . . .        0 0 0 0 0 0 0
0 0 0 1 0 0 0        . . . . . .        0 0 0 1 0 0 0
0 0 1 1 1 0 0        . . . . . .        0 0 1 1 1 0 0
0 . . . . . .    +   . 0 0 1 0 0 0  =   0 0 0 1 0 0 0
0 . . . . . .        . 0 1 1 0 0 0      0 0 1 1 0 0 0
0 . . . . . .        . 0 0 0 0 0 0      0 0 0 0 0 0 0
0 0 0 0 0 0 0        . . . . . .        0 0 0 0 0 0 0
```

So far, the result for erosion has not been defined round the edges of an image where any part of the structuring element falls outside the limits of the source picture. By default, the process of calculating the intersection of the translates of the source image simply omits any undefined source pixels. If all the source pixels which contribute to the result for a particular output pixel are undefined, the output pixel defaults to 1.

As an alternative, the **source** option allows you to specify a value for any pixel positions outside the **source** image which contribute to a translate of the source image. You specify the actual value with the **edge** key. All non-zero values for the **edge** key imply an edge value of 1. For erosion, an edge value of zero effectively closes off all foreground regions which touch the edge of the image. A non-zero edge value leaves regions "open" – it is as if regions extend indefinitely at right angles to the edges of the image. The combination **source edge 1** in fact produces the same result as the default case.

For example,

```
0 1 0 0 0                    0 1 0 0 0        E E E E E        0 1 1 0 0
0 1 1 0 0      1 0           0 1 1 0 0        E 0 1 0 0        0 1 1 1 0
0 1 1 1 0  (-) 0[1]    =     0 1 1 1 0   (&)  E 0 1 1 0   (&)  0 1 1 0 1
0 1 1 0 1      0 1           0 1 1 0 1        E 0 1 1 1        0 0 0 1 1
0 0 0 1 1                    0 0 0 1 1        E 0 1 1 0        E E E E E

                             0 E 0 0 0
                             0 0 1 0 0
                         =   0 0 1 0 0
                             0 0 0 0 1
                             0 0 0 E 0
```

where E represents the source edge value.

Instead of using the **source** option, you can, with the **output** option and the **edge** key, specify the value to assign to output pixels wherever the result depends on undefined source pixels. A zero output edge value is particularly useful when using erosion for templating operations. It eliminates possible edge effects because output pixels can only be set if the structuring element lies fully inside the source image. A non-zero pixel in the output image means that the structuring element exactly macthed the source image at that position.

berode

For example,

```
0 1 0 0 0                E E E E E
0 1 1 0 0      1 0       E 0 1 0 0
0 1 1 1 0  (-) 0[1]  =   E 0 1 0 0
0 1 1 0 1      0 1       E 0 0 0 1
0 0 0 1 1                E E E E E
```

where E represents the output edge value.

Notes

see also: **bdilate, bhmt, erode**

Defaults and Ranges

keys/options	defaults	range
[from]	current picture, held in the variable *select*	valid picture number
[to]	source picture	valid picture number
with	none	valid picture number
mask	none	valid picture number
times	1	postive integer
edge	0	real number
source/output	do not process undefined pixels	