NAME

pnmscale - scale a portable anymap

SYNOPSIS

pnmscale *scale_factor* [*pnmfile*]
pnmscale -reduce *reduction_factor* [*pnmfile*]
pnmscale [{**-xsize**=*cols* | **-width**=*cols* | **-xscale**=*factor*}] [{**-ysize**=*rows* | **-height**=*rows* | **-yscale**=*factor*}] [*pnmfile*]
pnmscale -xysize *cols rows* [*pnmfile*]
pnmscale -pixels *n* [*pnmfile*]

Miscellaneous options:
-verbose -nomix

Minimum unique abbreviation of option is acceptable. You may use double hypens instead of single hyphen to denote options. You may use white space in place of the equals sign to separate an option name from its value.

DESCRIPTION

Reads a PBM, PGM, or PPM image as input, scales it by the specified factor or factors and produces a PGM or PPM image as output. If the input file is in color (PPM), the output will be too, otherwise it will be grayscale (PGM). This is true even if the input is a black and white bitmap (PBM), because the process of scaling can turn a combination of black and white pixels into a gray pixel.

If you want PBM output, use **pgmtopbm** to convert **pnmscale**'s output to PBM. Also consider **pbmreduce**.

You can both enlarge (scale factor > 1) and reduce (scale factor < 1).

When you specify an absolute size or scale factor for both dimensions, **pnmscale** scales each dimension independently without consideration of the aspect ratio.

If you specify one dimension as a pixel size and don't specify the other dimension, **pnmscale** scales the unspecified dimension to preserve the aspect ratio.

If you specify one dimension as a scale factor and don't specify the other dimension, **pnmscale** leaves the unspecified dimension unchanged from the input.

If you specify the *scale_factor* parameter instead of dimension options, that is the scale factor for both dimensions. It is equivalent to **-xscale**=*scale_factor* **-yscale**=*scale_factor* .

Specifying the **-reduce** *reduction_factor* option is equivalent to specifying the *scale_factor* parameter, where *scale_factor* is the reciprocal of *reduction_factor*.

-xysize specifies a bounding box. **pnmscale** scales the input image to the largest size that fits within the box, while preserving its aspect ratio.

-pixels specifies a maximum total number of output pixels. **pnmscale** scales the image down to that number of pixels. If the input image is already no more than that many pixels, **pnmscale** just copies it as output; **pnmscale** does not scale up with **-pixels**.

If you enlarge by a factor of 3 or more, you should probably add a *pnmsmooth* step; otherwise, you can see the original pixels in the resulting image.

When the scale factor is not an integer (including all cases of scaling down), there are two ways to do the scaling. Which one**pnmscale** does is controlled by its **-nomix** option.

By default, **pnmscale** mixes the colors of adjacent pixels to produce output pixels that contain information from multiple input pixels. This makes the image look more like it would if it had infinite resolution. Note that it means the output may contain colors that aren't in the input at all.

But if you specify **-nomix**, **pnmscale** never mixes pixels. Each output pixel is derived from one input pixel. If you're scaling up, pixels get duplicated. If you're scaling down, pixels get omitted. Note that this means the image is rather distorted. If you scale up by 1.5 horizontally, for example, the even numbered input pixels are doubled in the output and the odd numbered ones are copied singly.

When the scale factor is an integer (which means you're scaling up), the **-nomix** option has no effect -- output pixels are always just N copies of the input pixels. In this case, though, consider using**pamstr etch** instead of **pnmscale** to get the added pixels interpolated instead of just copied and thereby get a smoother enlargement.

pnmscale with **-nomix** is faster than without, but **pnmenlarge** is faster still. **pnmenlar ge** works only on integer enlargements.

A useful application of **pnmscale** is to blur an image. Scale it down (without **-nomix)** to discard some information, then scale it back up using **pamstretch**.

Or scale it back up with **pnmscale** and create a "pixelized" image, which is sort of a computer-age version of blurring.

PRECISION

pnmscale uses floating point arithmetic internally. There is a speed cost associated with this. For some images, you can get the acceptable results (in fact, sometimes identical results) faster with **pnmscalefixed**, which uses fixed point arithmetic. **pnmscalefixed** may , however, distort your image a little. See **pnmscalefixed**'s man page for a complete discussion of the difference.

SEE ALSO

pnmscalefixed(1), **pamstretch**(1), **pbmreduce**(1), **pnmenlarge**(1), **pnmsmooth**(1), **pnmcut**(1), **pnm(5)**

AUTHOR