Pamarith User Manual

pamarith

Updated: 08 October 2005
Table Of Contents

NAME

pamarith - perform arithmetic on two Netpbm images

SYNOPSIS

pamarith -add | -subtract | -multiply | -divide | -difference | -minimum | -maximum | -mean | -compare | -and | -or | -nand | -nor | -xor | -shiftleft | -shiftright *pamfile1 pamfile2*

All options can be abbreviated to their shortest unique prefix. You may use two hyphens instead of one. You may separate an option name and its value with white space instead of an equals sign.

DESCRIPTION

This program is part of Netpbm.

pamarith reads two PBM, PGM, PPM, or PAM images as input. It performs the specified binary arithmetic operation on their sample values and produces an output of a format which is the more general of the two input formats. The two input images must be of the same width and height. The arithmetic is performed on each pair of identically located tuples to generate the identically located tuple of the output.

For the purpose of the calculation, it assumes any PBM, PGM, or PPM input image is the equivalent PAM image of tuple type **BLACKANDWHITE**, **GRAYSCALE**, or **RGB**, respectively, and if it produces a PBM, PGM, or PPM output, produces the equivalent of the PAM image which is the result of the calculation.

The first *pamfile* argument identifies the "left" argument image; the second *pamfile* argument identifies the "right" one.

If the output is PAM, the tuple type is the same as the tuple type of the left input image.

pamarith performs the arithmetic on each pair of identically located tuples in the two input images.

The arithmetic operation is in all cases fundamentally a function from two integers to an integer. The operation is performed on two tuples as follows. The two input images must have the same depth, or one of them must have depth one. **pamarith** fails if one of these is not the case.

If they have the same depth, **pamarith** simply carries out the arithmetic one sample at a time. I.e. if at a particular position the left input image contains the tuple (s1,s2,...,sN) and the right input image contains the tuple (t1,t2,...tN), and the function is f, then the output image contains the tuple (f(s1,t1),f(s2,t2),...,f(sN,tN)).

If one of the images has depth 1, the arithmetic is performed between the one sample in that image and each of the samples in the other. I.e. if at a particular position the left input image contains the tuple (s) and the right input image contains the tuple (t1,t2,...tN), and the function is f, then the output image contains the tuple (f(s,t1),f(s,t2),...,f(s,tN)).

Maxval

The meanings of the samples with respect to the maxval varies according to the function you select.

In PAM images in general, the most usual meaning of a sample (the one that applies when a PAM image represents a visual image), is that it represents a fraction of some maximum. The maxval of the image corresponds to some maximum value (in the case of a visual image, it corresponds to "full intensity."), and a sample value divided by the maxval gives the fraction.

For **pamarith**, this interpretation applies to the regular arithmetic functions: **-add**, **-subtract**, **-multiply**, **-divide**, **-difference**, **-minimum**, **-maximum**, **-mean**, and **-compare**. For those, you should think of the arguments and result as numbers in the range [0,1). For example, if the maxval of the left argument image is 100 and the maxval of the right argument image is 200 and the maxval of the output image is 200, and the left sample value in an **-add** calculation is 50 and the right sample is 60, the actual calculation is 50/100 + 60/200 = 160/200, and the output sample value is 160.

For these functions, **pamarith** makes the output image's maxval the maximum of the two input maxvals, except with **-compare**, where **pamarith** uses an output maxval of 2.

If the result of a calculation falls outside the range [0, 1), **pamarith** clips it -- i.e. considers it to be zero or 1-.

In many cases, where both your input maxvals are the same, you can just think of the operation as taking place between the sample values directly, with no consideration of the maxval except for the clipping. E.g. an **-add** of sample value 5 to sample value 8 yields sample value 13.

But with **-multiply**, this doesn't work. Say your two input images have maxval 255, which means the output image also has maxval 255. Consider a location in the image where the input sample values are 5 and 10. You might think the multiplicative product of those would yield 50 in the output. But **pamarith** carries out the arithmetic on the fractions 5/255 and 10/255. It multiplies those together and then rescales to the output maxval, giving a sample value in the output PAM of 50/255 rounded to the nearest integer: 0.

With the bit string operations, the maxval has a whole different meaning. The operations in question are: **-and**, **-or**, **-nand**, **-nor**, **-xor**, and **-shiftleft**, **-shiftright**.

With these, each sample value in one or both input images, and in the output image, represents a bit string, not a number. The maxval tells how wide the bit string is. The maxval must be a full binary count (a power of two minus one, such as 0xff) and the number of ones in it is the width of the bit string. For the dyadic bit string operations (that's everything but the shift functions), the maxvals of the input images must be the same and **pamarith** makes the maxval of the output image the same.

For the bit shift operations, the output maxval is the same as the left input maxval. The right input image (which contains the shift counts) can have any maxval and the maxval is irrelevant to the interpretation of the samples. The sample value is the actual shift count. But it's still required that no sample value exceed the maxval.

The Operations

Most of the operations are obvious from the option name.

-subtract subtracts a value in the right input image from a value in the left input image.

-difference calculates the absolute value of the difference.

-multiply does an ordinary arithmetic multiplication, but tends to produce nonobvious results because of the way **pamarith** interprets sample values. See Maxval.

-divide divides a value in the left input image by the value in the left input image. But like **-multiply**, it tends to produce nonobvious results. Note that **pamarith** clipping behavior makes this of little use when the left argument (dividend) is greater than the right argument (divisor) -- the result in that case is always the maxval. If the divisor is 0, the result is the maxval. This option was new in Netpbm 10.30 (October 2005).

-compare produces the value **0** when the value in the left input image is less than the value in the right input image, **1** when the values are equal, and **2** when the left is greater than the right.

If the maxvals of the input images are not identical, **pamarith** may claim two values are not equal when in fact they are, due to the precision with which it does the arithmetic. However, it will never say A is greater than B if A is less than B.

-compare was new in Netpbm 10.13 (December 2002).

-and, **-nand**, **-or**, **-nor**, and **-xor** consider the input and output images to contain bit strings; they compute bitwise logic operations.

-shiftleft and **-shiftright** consider the left input image and output image to contain bit strings. They compute a bit shift operation, with bits falling off the left or right end and zeroes shifting in, as opposed to bits off one end to the other. The right input image sample value is the number of bit positions to shift.

Note that the maxval (see Maxval) determines the width of the frame within which you are shifting.

Notes

If you want to apply a unary function, e.g. "halve", to a single image, use **pamfunc**.

SEE ALSO

pamfunc, **pamsummcol**, **pamsumm**, **pnminvert**, **ppmbrighten**, **ppmdim**, **pnmconvol**, **pamdepth**, **pnmpsnr**, pnm, pam

HISTORY

pamarith replaced **pnmarith** in Netpbm 10.3 (June 2002).

In Netpbm 10.3 through 10.8, though, **pamarith** was not backward compatible because it required the input images to be of the same depth, so you could not multiply a PBM by a PPM as is often done for masking. (It was not intended at the time that **pnmarith** would be removed from Netpbm -- the plan was just to rewrite it to use **pamarith**; it was removed by mistake).

But starting with Netpbm 10.9 (September 2002), **pamarith** allows the images to have different depths as long as one of them has depth 1, and that made it backward compatible with **pnmarith**.

The original **pnmarith** did not have the **-mean** option.

The **-compare** option was added in Netpbm 10.13 (December 2002).

The bit string operations were added in Netpbm 10.27 (March 2005).

The **-divide** option was added in Netpbm 10.30 (October 2005).

Table Of Contents

NAME
SYNOPSIS
DESCRIPTION
MAXVAL
THE OPERATIONS
NOTES
HISTORY
SEE ALSO
AUTHOR