images/cc

Orfeo ToolBox users meeting and hackfest 2015

Third parties policy and SuperBuild

OTB development team

3 - 5 june 2015, Toulouse

Variables

- im1 = a pixel from first input, made of n components (n bands) = Vector
- im1bj = jth component of a pixel from first input (first band is indexed by 1) = Scalar
- im1PhyX and im1PhyY = spacing of first input in X and Y directions (horizontal and vertical) = Scalar
- idxX and idxY = represent the indices of the current pixel (scalars) = Scalar
- im1bjMean im1bjMin im1bjMax im1bjSum im1bjVar = mean, min, max, sum, variance of jth band from first input (global statistics) = Scalar
- im1bjNkxp = a neighbourhood ('N') of pixels of the jth component from first input, of size kxp = Matrix

.	.	.
.	.	.
.	.	.
.	.	.
.	.	.

Neighborhood of 3x5. k/p = horizontal/vertical direction. k and p must be odd numbers.

Some examples 1

- Always keep in mind that a pixel of an otb::VectorImage is always represented as a row vector inside the muParserX framework
- MuParserX only addresses mathematically well-defined formulas

Formula	Status
im1 + im2	correct only if the two first inputs have the same number of bands (No
im1 + 1	incorrect even if im1 represents a one-band pixel
im1 + {1}	much better !
im1 + {1,1,1,...,1}	correct if im1 is made of n bands

Some examples 2

- Always keep in mind that a pixel of an otb::VectorImage is always represented as a row vector inside the muParserX framework
- MuParserX only addresses mathematically well-defined formulas

Formula	Status
im1b1 + 1	correct
{im1b1} + {1}	correct
im1b1 + {1}	incorrect
{im1b1} + 1	incorrect
im1 + {im2b1,im2b2}	correct if im1 represents a pixel of two components

Some examples 3

- Always keep in mind that a pixel of an otb::VectorImage is always represented as a row vector inside the muParserX framework
- MuParserX only addresses mathematically well-defined formulas

Formula	Status
{im2b1,im2b2}*{1,2}	incorrect
{im2b1,im2b2}*{1,2}'	correct
im2*{1,2}'	correct if im2 represents a pixel of two components

New operators and functions

New operators and functions have been implemented within BandMathX application. These ones can be divided into two categories.

- adaptation of existing operators/functions, that were not originally defined for vectors and matrices (for instance cos, sin, ...). These new operators/ functions keep the original names to which we add the prefix "v" for vector (vcos, vsin, ...).
- truly new operators/functions.

New operators and functions

- div (element-wise division) and dv (division by a scalar)
- mult (element-wise multiplication) and mlt (multiplication by a scalar)
- mult (element-wise exponentiation) and mlt (exponentiation by a scalar)

Operator/function	ex. 1			ex. 2		
div and dv	im1	div	im2	m1	dv	2.0
mult and mlt	im1	mult	im2	im1	mlt	2.0
pow and pw	im1	pow	im2	im1	pw	2.0

New operators and functions

  - dotpr : This function allows the dot product between two vectors or matrices (actually in our case, a kernel and a neighbourhood of pixels) $\sum_{(i,j)} m_1(i,j) * m_2(i,j)$
- For instance: dotpr(kernel1,im1b1N3x5) is correct provided that kernel1 and im1b1N3x5 have the same dimensions.
- The function can take as many neighbourhoods as needed in inputs. Thus, if n neighbourhoods must be processed, the output will consist in a row vector of n values. This behaviour is typical of the functions implemented in the BandMathX application.

New operators and functions

- mean : mean value of a given vector or neighborhood
- var : variance value of a given vector or neighborhood
- median : median value of a given vector or neighborhood
- corr : correlation between two vectors or matrices of the same dimensions (the function takes two inputs)
- maj : compute the most represented element within a vector or a matrix
- vmin and vmax : min or max value of a given vector or neighborhood

Operator/function	example
mean (*)	mean(im1b1N3x3,im1b2N3x3,im1b3N3x3,im1b4N3x3)
var (*)	var(im1b1N3x3)
median (*)	median(im1b1N3x3)
corr (two inputs)	corr(im1b1N3x3,im1b2N3x3)
maj (*)	maj(im1b1N3x3,im1b2N3x3)
vmin and vmax (one input)	(vmax(im3b1N3x5)+vmin(im3b1N3x5)) div {2.0}

(*) : the function can take as many inputs as needed; one mean value is computed per input

New operators and functions

- car : This function allows to concatenate the results of several expressions into a multidimensional vector, whatever their respective dimensions (the function can take as many inputs as needed)
- band : This function allows to select specific bands from an image, and/or to rearrange them in a new vector.

Operator/function	example
cat	cat(im3b1,vmin(im3b1N3x5),median(im3b1N3x5),vmax(im3b1N3x5))
band	bands(im1,{1,2,1,1})

Note about cat function : the user should prefer the use of semi-colons (;) when setting expressions, instead of directly use this function. The application will call the function 'cat' automatically.

Filter : example 1

  - include "otbBandMathXImageFilter.h"
  - ....
  - typedef otb::BandMathXImageFilter¡ImageType¿ FilterType;
  - ...
  - FilterType::Pointer filter = FilterType::New();
  - ...
  - filter- *SetExpression*(" *im*1 − *mean*(*im*1*b*1*N*5*x*5, *im*1*b*2*N*5*x*5, *im*1*b*3*N*5*x*5, *im*1*b*4*N*5*x*5)"); *filter* − *SetNthInput*(0, *reader*− > *GetOutput*()); *oufilter*− > *SetNthInput*(0, *reader*− > *GetOutput*(), " *imageA*");
- writer- *SetInput*(*filter*− > *GetOutput*()); *writer* − *Update*();

Filter : example 2

  - filter- $SetMatrix(''kernel'', ''0.1, 0.1, 0.1; 0.1, 0.2, 0.1; 0.1, 0.1, 0.1''); filter - SetConstant(''cst'', 1.0);$

- filter- $SetExpression(''bands(im1, 1, 2, 3) - dotpr(kernel, im1b1N3x3, im1b2N3x3, im1b3N3x3) + cst, cst, cst''); filter - ExportContext(argv[4]);$
- filter- $ImportContext(argv[4]);$
  Note : concatenation of the results of several expressions into a multidimensional vector is possible. For this purpose, use semi-colons (;) as separators between expressions.

Context

F expo 1.1 M kernel1 0.1 , 0.2 , 0.3; 0.4 , 0.5 , 0.6; 0.7 , 0.8 , 0.9; 1 , 1.1 , 1.2; 1.3 , 1.4 , 1.5 E cat(dotpr(kernel1,imageAb1N3x5,imageAb2N3x5), im2b1$^{e}xpo, vcos(canal3), mean(imageAb2N3x3), var(imageAb2N3x3), median(imageAb2N3x$