# Comments on the VSIPL *Image Processing Operations* definition

## *Prolog*

As part of the Cameron project, we have recently developed a complete implementation of the proposed VSIPL Image Processing Operations. We are not putting this implementation forth as a reference implementation because its correctness has not been verified, and because it was written in an experimental language called SA-C. (See http://www.cs.colostate.edu/cameron/ for a description of SA-C and its role in programming adaptive computing systems). Nonetheless, in the process of implementing the VSIPL IP Operations in SA-C, we encountered a number of problems with the proposed standard that we think should be discussed by the VSIPL IP community, and if possible corrected.

Let us emphasize that the following is not meant as personal criticism. The authors of the IP manual had a daunting task: to write a complex standard in a short amount of time with relatively little feedback. I hope these comments will be taken as constructive criticism, and not as an affront to the authors. We simply hope to provide comments that might help refine the standard before it is adopted.

These comments are in response to version 0.2 of the *Image Processing Operations* manual, dated April 28, 1998, and accessible through http://www.vsipl.org/

## *Introduction*

While implementing the VSIPL Image Processing Operations standard, we encountered a number of inconsistencies, errors and omissions that we feel need to be addressed. In general, we will avoid the question of whether the standard is complete, and what procedures might be added to it. We assume that the standard can be extended over time. Instead, we will focus on technical errors within the existing specification. These errors can be grouped roughly as follows:

1. Complex data type errors.
2. Range Errors
3. Arithmetic errors.
4. Indexing errors.
5. Off-by-one errors (easily fixed)
6. Window size errors (easily fixed)

Finally, before we begin, we will violate our own rule on questioning whether the standard is complete only once in passing: since the discrete cosine transform is in the standard, shouldn't the inverse discrete cosine transform be included also?

## *Complex Data Type Errors*

The VSIPL Image Processing Operations standard defines many functions over images with complex pixels that cannot logically be implemented. In almost all cases, the problem reduces to assuming (falsely) that complex number can be ordered, and therefore that it is possible to take the minimum, maximum or median of a set of complex values. Complex numbers, of course, involve a pair of values in orthogonal dimensions, and while it is possible to take the minimum, maximum or median of the magnitudes of a set of complex numbers, no ordering over the complex numbers as a whole is defined. For this reason, the VSIPLS Scalar Functions manual (version 0.01, dated Sept. 14, 1997) does not define complex minimum, complex maximum, or complex median operators.

It is possible to interpret these functions as operating over the magnitudes of the complex pixels, and for our prototype implementation this is what we did. This "solution" creates ambiguities, however. If a set of pixels contains both 1+10i and 10+1i, which one is the maximum? (Their magnitudes are equal.) In our implementation, the first maximum or minimum value is used, but this is not a good situation (just try to predict the value of the median!).

Logically, there is another possible "solution": the minimum (maximum, median) of a set of complex numbers could be the minimum (maximum, median) *magnitude* of the numbers. This would imply, however, that the minimum (maximum, median) of a set of complex numbers is not a complex number itself. This is conceptually ugly, in part because it implies that the output data types don't match the input data types for many functions.

As a result of this problem, the following VSIPL Image Processing operators are not well defined:

**vsip_cidilate_i**
**vsip_cidilate_f**
**vsip_cierode_i**
**vsip_cierode_f**
**vsip_ciresample_i**
**vsip_ciresample_f**
**vsip_cistats_i**
**vsip_cistats_f**

In addition, the morphological open and close operators are defined in terms of dilate and erode, so they are indirectly not well defined.

## *Range Errors*

The proposed VSIPL Image Processing Operations manual defines two versions of the Discrete Cosine Transform (DCT) whose output ranges are problematic. One version defines both the input and output images to be byte. Unfortunately, the DCT converts an image from the spatial domain to the frequency domain, and the range of values in the frequency domain is much larger than in the spatial domain. Also, frequency values are signed values that are equally likely to be positive or negative. As a result, it is virtually impossible to find a byte image whose frequency values (after DCT) will not overflow or underflow a byte output image.

In our opinion, when the input image has byte pixels, the output image should have int or float pixels (int if the DCT is being used for compression; float otherwise). If the inverse DCT is added to the standard – and it should be – it should have versions that take int and float input and produces byte output. The current version for byte images is not useful.

Taking the DCT of a binary image is an odd thing to do. (There are better ways of compressing binary images.) Oddity aside, however, the DCT of a binary image as defined has the same problem that the byte version does, namely that the output absolutely needs to be signed, and probably needs a larger range. Since there is no signed byte data type in VSIPL, this implies that the result of applying DCT to a bit image must by an int or float image. (Note, however, that the introduction to the manual suggests a signed short type.)

As a result of this problem, the following operators cannot be used:

**vsip_idct8x8_bit**
**vsip_idct8x8_byte**

## *Arithmetic Errors*

The proposed VSIPL Image Processing Operations manual does not describe what to do in the case of arithmetic exceptions. In particular, it does not define what to do in the case of overflow and underflow. As a rule, this is not critical when the data types in question are int or float (either real or complex), and it doesn't arise at all when the data type is binary (bit). What to do on overflow is absolutely critical, however, when it comes to byte operations. Note that overflows and underflows are common during convolution, including edge detection (which as defined is a special case of convolution).

There are three possible defaults. The most obvious is to signal an error and abort, but this is generally not desirable. The second is to drop the most significant bits and "wrap around" (as C would do); this is an even worse option, since the most significant edges (those with the highest contrast) will get lost. The best solution is the solution adopted by the Intel Image Processing Library: define your operations to be saturating, so that overflow results in the maximum possible value and underflow results in the minimum possible value.

Until this issue is resolved, the following operators in particular are not well defined:

**vsip_iconv_byte**
**vsip_isobel_byte**
**vsip_iroberts_byte**
**vsip_iprewitt_byte**

## *Indexing Errors*

The function for equalizing an image's binned histogram is defined in terms of an index that will create an out of bounds error. In particular, it is defined as the sum from r=0 to P of P(r)+wP(r+1). (The weighting term w is a complex equation that is not relevant to this discussion.) The problem, of course, is when r=P (the max value). In this case, the equation tries to reference histogram bin P+1, which is out of bounds.

This error affects:

**vsip_ihisteq_byte**
**vsip_ihisteq_i**
**vsip_ihisteq_f**

## *Off by One Errors (Easily Fixed)*

The histogram functions in the proposed VSIPL Image Processing Operations manual depend on the range of the input pixels, which they define as ($src_{max} - src_{min}$). While this is correct when the input pixels are of type float (the range from 2.2 to 1.0 is 1.2), it is incorrect when the input pixels are of type int. For integer pixels, the range ought to be (max-min+1). This easily fixable error infects the following definitions:

**vsip_icdfeq_i**
**vsip_ihisteq_byte**
**vsip_ihisteq_i**

## *Window Size Errors (Easily Fixed)*

Some errors begin life as a typographical error, only to be spread via "cut and paste" until they become systemic. Such may be the origin of the repeated error with regard to window sizes.

According to the VSIP IP Operatons manual, the convolution operator convolves an image with a kernel of size p, producing an output image that is the same size as the source image, or that is larger or smaller than the source image in each dimension by (p-1)/2. This seems unlikely. For example, if the kernel is 3x3, and the output image is only defined where the kernel fits entirely within the source image, then the image will shrink along each edge by (p-1)/2 = 1 pixel. There are two edges in each dimension, however (top + bottom, left + right), so the output image is smaller by 2*((p-1)/2) = p-1 pixels. Similarly, if the output is defined anywhere the kernel overlaps the source image, then it will grow by p-1 pixels in each dimension (not [p-1]/2, as stated in the manual). This error is in the "restrictions" portion of the following definitions:

**vsip_iconv_xxx**
**vsip_ciconv_xxx**
**vsip_idilate_xxx**
**vsip_cidilate_xxx**
**vsip_ierode_xxx**
**vsip_cierode_xxx**
**vsip_iopen_xxx**
**vsip_ciopen_xxx**
**vsip_iclose_xxx**
**vsip_ciclose_xxx**