BINARY MORPHOLOGY – ATOMIC OPERATIONS

Separable atomic operations with block Sels, Sel decomposition, sequence interpreters and simplicity of use

With all the mechanisms set up for doing binary morphology with both rasterop and dwa, and for doing grayscale morphology, it is important to add some machinery to make it very easy to use for the situation where the Sels are linear operators of all HITS, which is by far the most common usage. These linear operators can be used in "separable" combinations, horizontal and vertical, to implement morphology with 2-dimensional rectangular Sels of all HITS. We call these 2-dimensional Sels "bricks." Large linear Sels, of which the bricks are composed, can themselves be decomposed for efficiency, and we do this with 2-way composites, giving a computation savings for a linear Sel of length n of about a factor of n / (2 * sqrt(n)) over the *unary* method without decomposition.

The machinery has two levels that are convenient for use: a lower level where the brick Sel morphologyical operations are directly invoked, and a higher level that is an interpreter to run a sequence of such operations. Specifically,

1. Atomic functions for brick Sels, using separability and optionally 2-way composition.
2. A set of interpreters to implement a sequence of these atomic functions.

Atomic separable functions

There are five sets of atomic functions for brick Sels, all with similar interfaces:

- Grayscale morphological operations (in *graymorph.c*).
- Binary morphology using rasterops (the `pix*Brick()` functions in *morph.c*).
- Binary morphology using rasterops with 2-way Sel composition (the `pix*CompBrick()` functions in *morph.c*).
- Binary morphology using dwa (the `pix*BrickDwa()` functions in *morphdwa.c*).
- Binary morphology using dwa with 2-way Sel composition (the `pix*CompBrickDwa()` functions in *morphdwa.c*).

These all take the sizes of the horizontal and vertical dimensions of the structuring element, and do separable implementions when both dimensions are greater than 1. They are useful in their own right because they implement correct boundary conditions,

including safe closing if chosen. They also handle all intermediate images transparently, of which there are several for separable openings and closings. And they don't require the generation and destruction of Sels, as they make them internally and destroy them after use. See the notes at the beginning of *morph.c* for usage.

For the operations using 2-way composable Sels, consider `pixOpenCompBrick()` as an example. The results are identical to to the those of `pixOpenBrick()`, for sizes that are exactly decomposed such that the product of factors in the former equals the Sel size of the latter. We place constraints and penalties on the functions that choose how to do the decomposition. The regression test *prog/binmorph2_reg.c*demonstrates the result of decomposition. For large Sels, the `pix*CompBrick()` functions are much more efficient. *prog/binmorph2_reg* also tests the dwa composable Sel operations, such as`pixOpenCompBrickDwa()`.

The dwa brick morphological operations work on the set of linear Sels that are generated by `selaAddBasic()`, and which are implemented in dwa by the functions in *fmorphgen.1.c*. Look there to see which are available. If for some reason you want some other Sel, such as sel_23h, which is not there, you can follow one of the prescriptions set out in *morphdwa.c*. However, you will likely find that you don't need to do this, because the dwa operations that use 2-way composable Sels will implement a close approximation to sel_23h. For convenience, if you call the non-composite dwa version, such as`pixCloseBrickDwa()`, and the Sel doesn't exist for it, it will automatically call the 2-way composite dwa version, `pixCloseCompBrickDwa()`.

Interpreters for sequences of atomic operations

To make these atomic functions even simpler to use, we provide interpreters for sequences of morphological operations. For binary morphology, these sequences are combined with rank reductions and replicative expansion. All the interpreters are in *morphseq.c*. The morphological operations enabled by the sequences all use separable brick Sels. As with the atomic brick dwa functions, the interpreters reduce the complexity of using a dwa implementation to s single function invocation with a simple interface.

How do we test all this? A large regression test, *prog/binmorph1_reg.c*, tests all the binary brick operations, including the 2-way Sel decomposition, along with their interpreted sequences. Specifically, we test all morphological operations using:

- general rasterop
- brick rasterop (unary Sel)
- morph sequence of brick rasterop (unary Sel)
- morph sequence of composite brick rasterop

- brick dwa
- morph sequence of brick dwa
- morph sequence of composite brick dwa

A second regression test, *prog/binmorph2_reg.c*, compares the results of the 2-way composite separable brick implementations (both rasterop and dwa) against the unary brick implementations. A third regression test, *prog/binmorph3_reg.c*, compares rasterop brick (separable and non-separable) with various implementations of separable dwa bricks.

Our default boundary condition is asymmetric (0 for both dilation and erosion), but you can toggle this to test all functions with using the symmetric b.c. as well.

Source : http://www.leptonica.com/binary-morphology.html