SUMMARY OF AVAILABLE IMPLEMENTATIONS

A summary might help describe all these different implementations, and how they are used. The top-level for all these is in *morph.c* (full image rasterops) and *morphdwa.c* (destination word accumulation -- dwa).

Full image rasterops

- Full image rasterops works with any arbitrary Sel (including, of course, hit-miss Sels). Consider dilating an image with a 40 x 40 brick Sel. `pixDilate()` will require 1600 full page rasterops, Brute force.
- The next level up in complexity are the brick rasterops, which are separable. `pixDilateBrick()` will carry out the 40 x 40 dilation with a mere 80 rasterops.
- Next up in complexity are the two-way decomposable brick rasterops. These use two Sels, a linear Sel and a comb Sel. `pixDilateCompBrick` will perform the 40 x 40 dilation using a linear brick of length 8 and a 5-tine comb in each direction, for a total of 26 rasterops. Separability and decomposability are important for large Sels!

Destination word accumulation

Dwa implementations are typically about 3-5 times faster than full image rasterops. The implementation is perhaps more complicated, but no harder to use for the brick Sels.

- A general Dwa operation on an Sel of all hits or hits and misses can be made using the function `fmorphautogen()` or `fhmtautogen()`, respectively. These functions take an array of Sels (a Sela), and generate all the code required for any of the four basic morphological operations. Two interfaces are given: a higher one that adds and removes border pixels to avoid boundary effects, and a lower one that assumes the border pixels already exist. The first is useful for simple operations; the second for sequences where you don't want to be adding and removing borders multiple times. Once generated, the code can be compiled directly with an program that uses it, or it can be compiled into the library. Your choice.
- Special functions, such as `pixDilateBrickDwa()`, have been written for Dwa operations using separable brick Sels. Only a small number of linear Sels have been compiled in. However, for operations where the Sel doesn't exist, the brick Dwa operation defaults to the composite one (next).
- Composite separable functions, such as `pixDilateCompBrickDwa()`, have been written for all brick Sels up to 63 x 63. (Remember: a Dwa Sel cannot have hits or

misses more than 31 pixels on any side of the Sel origin, so putting the origin in the Sel center, this gives a maximum linear Sel size of 63.) Composable brick operations require a sequential operation using a small linear brick and and extended comb; the order does not matter. The characteristics of each are given in a struct in `sel1.c`. The identical combinations are used in the full image rasterop implementation, which is slower than the Dwa but has the ability to go to arbitrary sizes.

As you can see, much of the machinery is specifically targeted for brick Sels. The reason is that in most applications, the vast majority of morphological operations use brick Sels!

There are a set of 3 regression tests, *prog/dwamorph*_reg.c*, that show you how to generate and use new Sels. *prog/dwamorph1_reg.c* calls a function to generate all linear Sels (both horizontal and vertical) from lengths between 2 and 63. This is a lot of code to add to the library, so instead, it compiles and runs two other programs with the code linked in. These programs are:

1. *prog/dwamorph2_reg*. This runs a regression test comparing dwa to rasterop morphology for all the basic operations and all 122 Sels.
2. *prog/dwamorph3_reg*. This runs timing for linear and composable operations for both rasterop and dwa, for all the horizontal Sels. The results are shown at the end of this section.

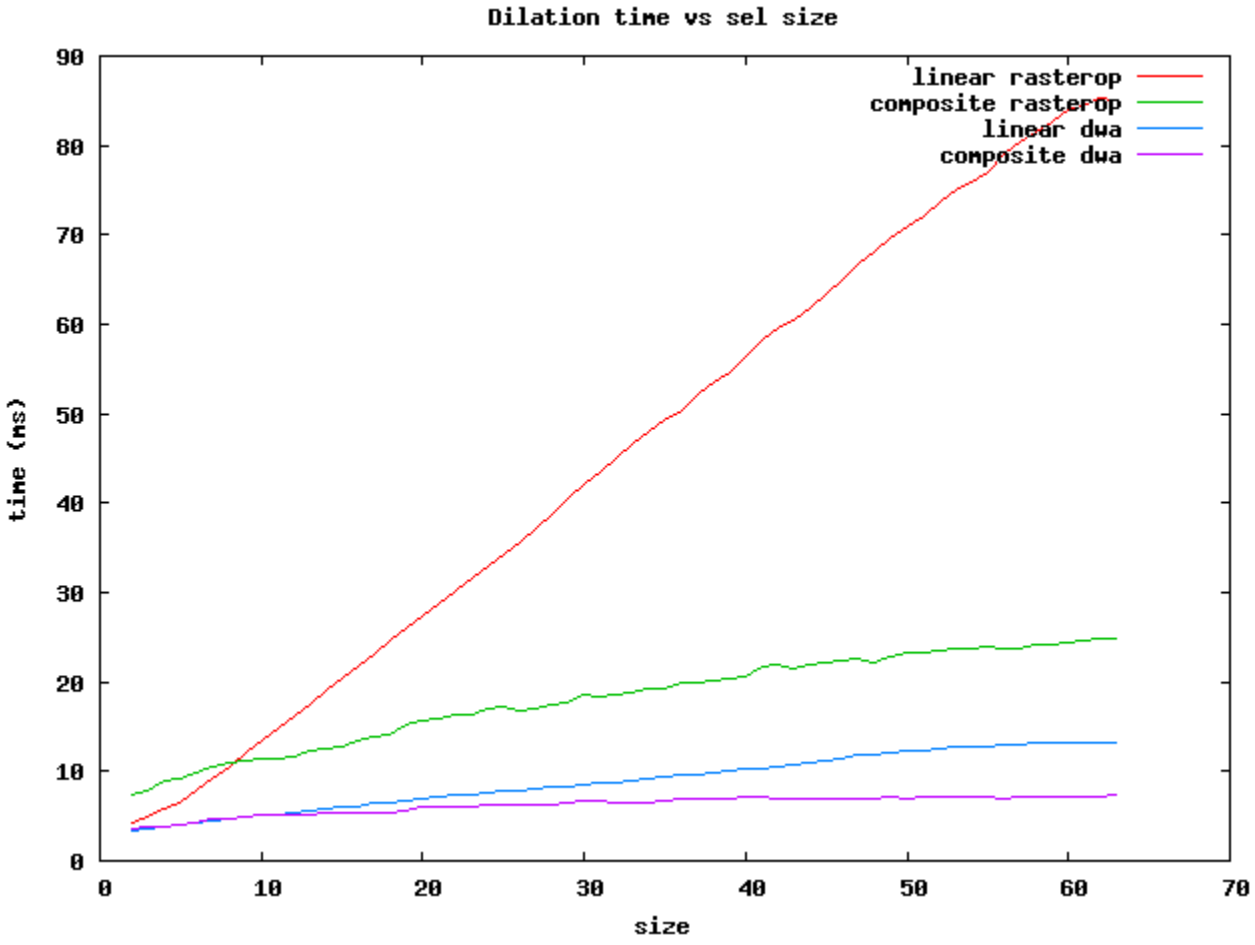
Sequence interpreters

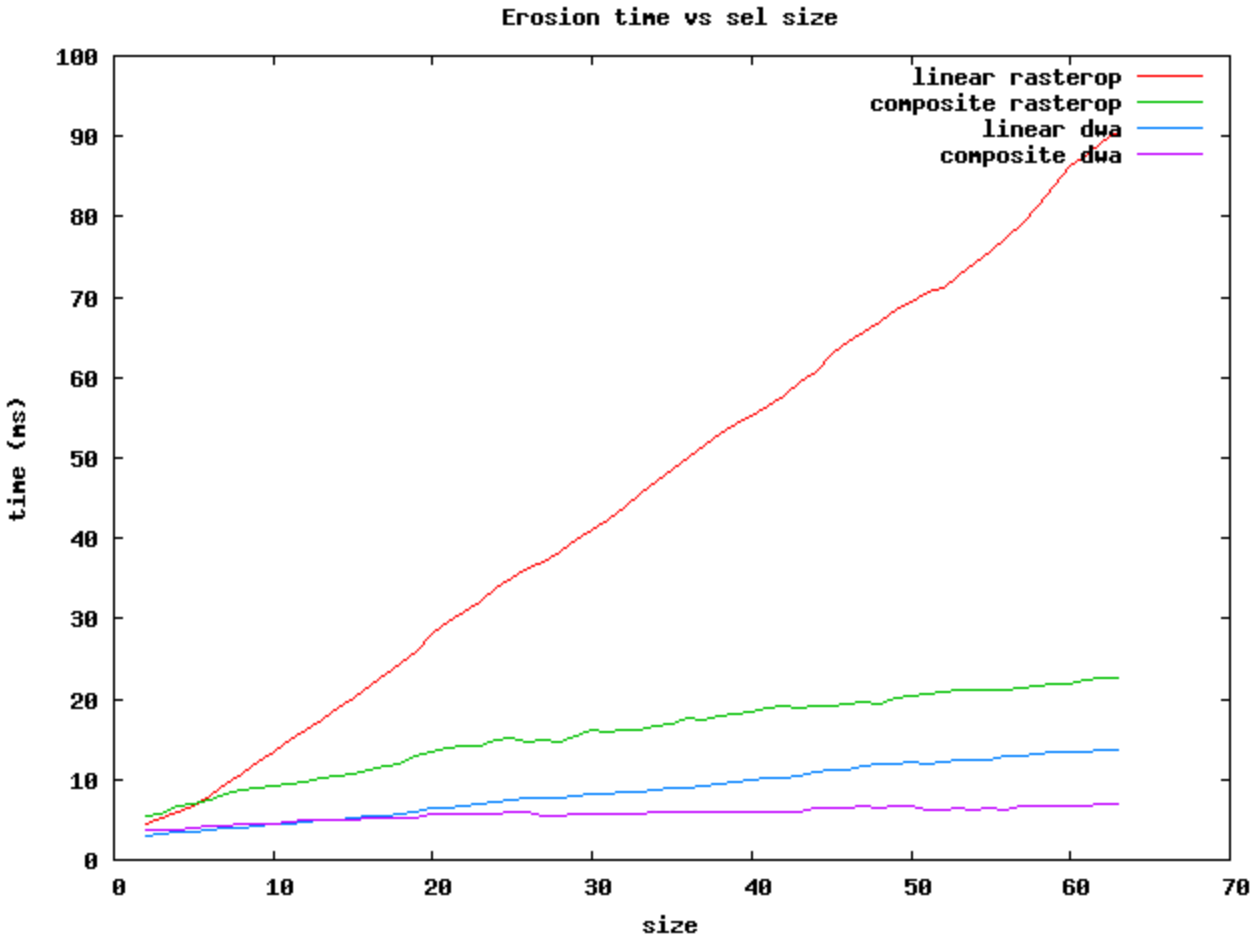
As mentioned above, we supply a number of interpreters in *morphseq.c* for carrying out these brick operations. The interpreters allow you to do a sequence of operations with a single command. They handle all intermediate images in the pipeline, simplifying the usage and avoiding errors in generating, using and destroying the intermediate images. We have interpreters for:

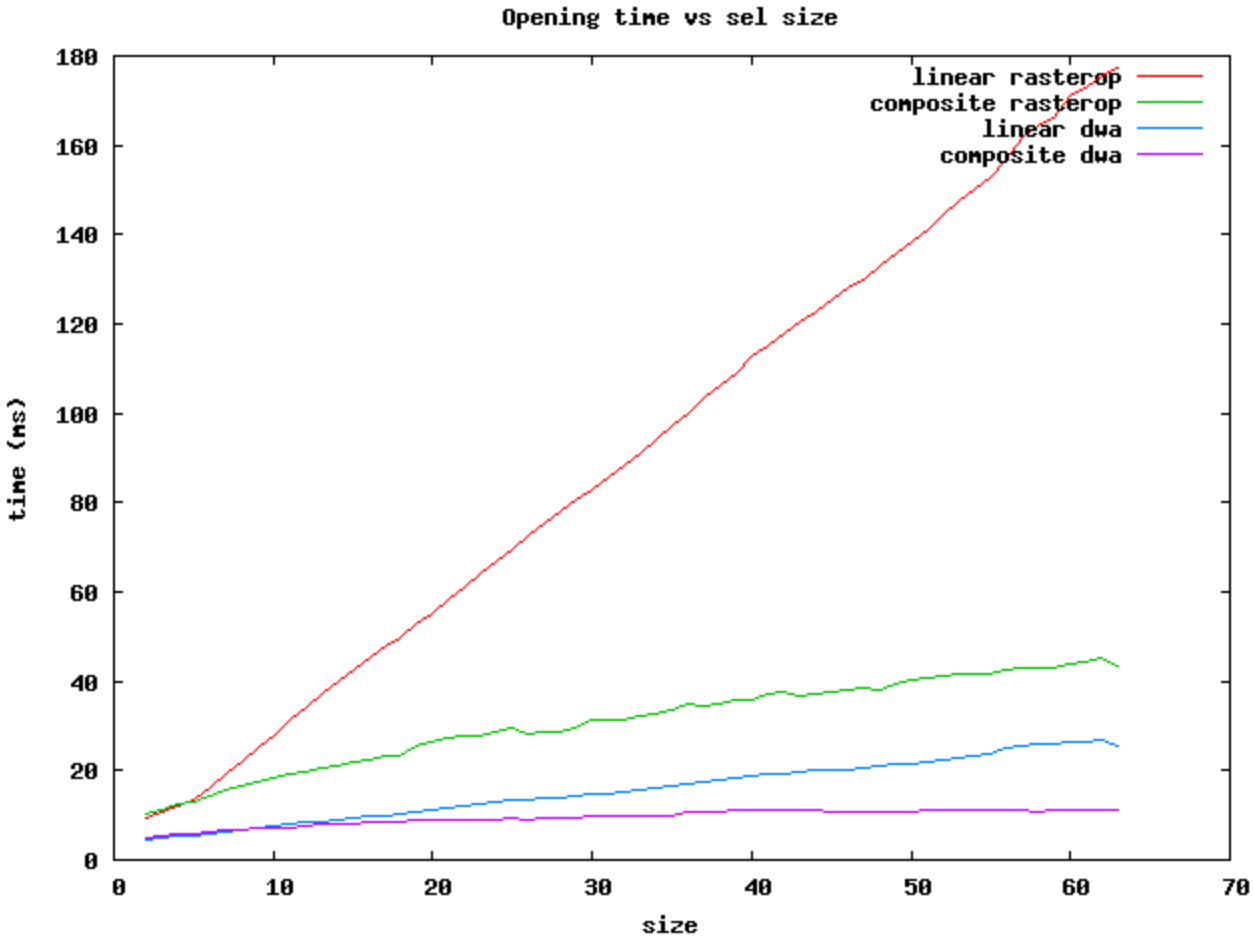
- *Rasterops on separable bricks.* These are run using `pixMorphSequence()`.
- *Rasterops on composable separable bricks.* These are run using `pixMorphCompSequence()`.
- *Dwa on separable bricks.* These are run using `pixMorphSequenceDwa()`.
- *Dwa on composable separable bricks.* These are run using `pixMorphCompSequenceDwa()`.

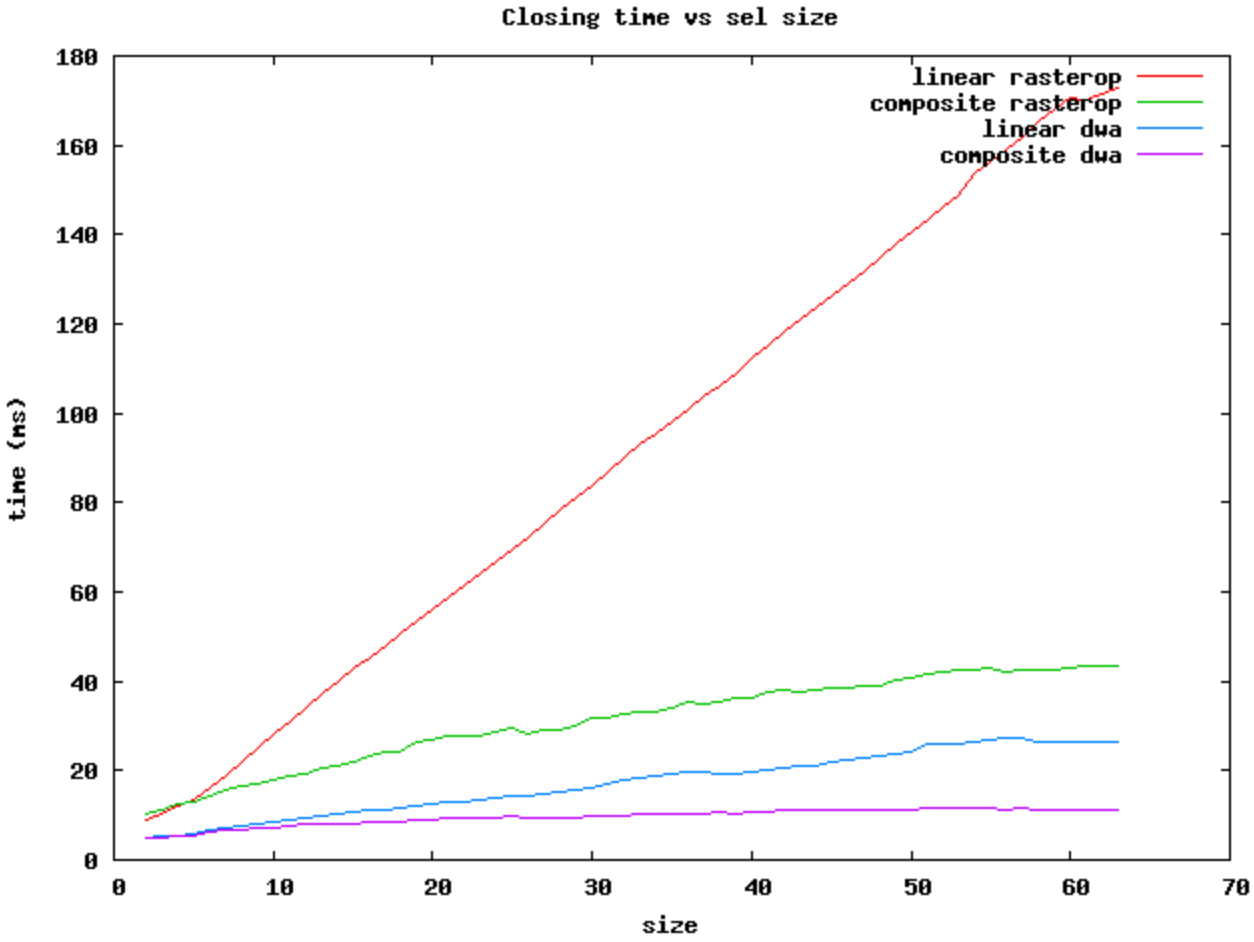
Performance comparisons

Performance of linear and composite operations for rasterops and dwa are given in the four plots below. The time for each is given in milliseconds for operations on an image (feyn.tif), which has about 8 million pixels (300 ppi scan on letter size paper). The operations are done on *horizontal* brick Sels of lengths between 2 and 63. (Operations on *vertical* brick Sels are considerably faster for rasterop, but only slightly faster for dwa.) This data was generated directly by *prog/dwamorph3_reg* on a 3 GHz P4, and should be representative for those machines.




Erosion time vs sel size
linear rasterop
composite rasterop
linear dwa
composite dwa
time (ms)
size
0
10
20
30
40
50
60
70
80
90
100


Opening time vs sel size
linear rasterop
composite rasterop
linear dwa
composite dwa
time (ms)
size
0
20
40
60
80
100
120
140
160
180
0
10
20
30
40
50
60
70



There are several things to notice:

- The composite morphological operation times scales approximately as the square root of the Sel size. The dilation starts above the others because it is necessary to add a border for dilation to avoid boundary effects.
- The speed of dwa is typically about 4 times faster than full image rasterops.
- The time to do a dwa operation can be estimated from the fact that dwa performs about 12 pixel operations for each machine cycle. For example, a linear erosion of size 60 on a 8 x 10^6 pixel image reuires about 5 x 10^8 pixel operations. A 3 GHz machine can perform about 4 x 10^10 pixel operations/sec, so the erosion should take about 13 msec. The comparable composite operation (6 + 10 = 60) is expected to take about 4 msec. As you can see from the plot, there is an additional 2 msec overhead for adding and removing a border, so the composite dwa erosion takes about 6 msec, and the opening and closing require an additional 4 msec (total: 10 msec).

In summary, for the fastest operations on brick Sels, use the dwa composite for sizes less than 64 and the rasterop composite for larger ones.

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