Description of the bmorph.m function

Tudor Dima, tudima@yahoo.com

January 27, 2010

Contents

List of Figures

1 Introduction

When using the existing matlab functions `imdilate.m` and `imerode.m` with anything else than the 'square' and 'disk' options one notices a huge performance drop. Moreover, the 'disk' structuring elements are actually octagonal, making the addition of a true 'circular' element a necessity.

A new function has been written in order to improve the speed of the binary morphological functions provided with the Matlab Image Processing Toolbox. Also, its architecture allows the addition of an extension that will allow iterative calls using increasingly larger structuring elements on the same original image. This extension is used when sweeping the size of the structuring element while eroding/dilating the same original image.

This new function will replace the `imerode.m` and `imdilate.m` functions when the structuring element is anything else than 'square' or 'disk' or in case of repeated calls (for instance when sweeping the structuring element while looking for a certain effect, threshold in image statistics, etc).

2 Description of the function `bmorph.m`

2.1 Matlab syntax, examples

The first call when applying erosion is :

```
[IMG_eroded, Contours] = bmorph(IMG, B_prev, 0, [], 1);
```

this will do a first erosion (including edges) using element `B_prev` and will also generate the contours, to be used in the subsequent calls. These calls will be of the form:

```
IMG_eroded = bmorph(IMG_eroded,{B, B_prev}, 0, Contours, 0);
```

Executing this will further erode `IMG_eroded` with only the pixels in `B` that do not belong to `B_prev`, all while re-using the `Contours`, calculated only once.

Similarly, for dilation the first call is:

```
[IMG_dilated, Contours] = bmorph(IMG, B_prev, 1);
```

and the subsequent ones:

```
IMG_dilated = bmorph(IMG_dilated,{B, B_prev}, 1, Contours);
```

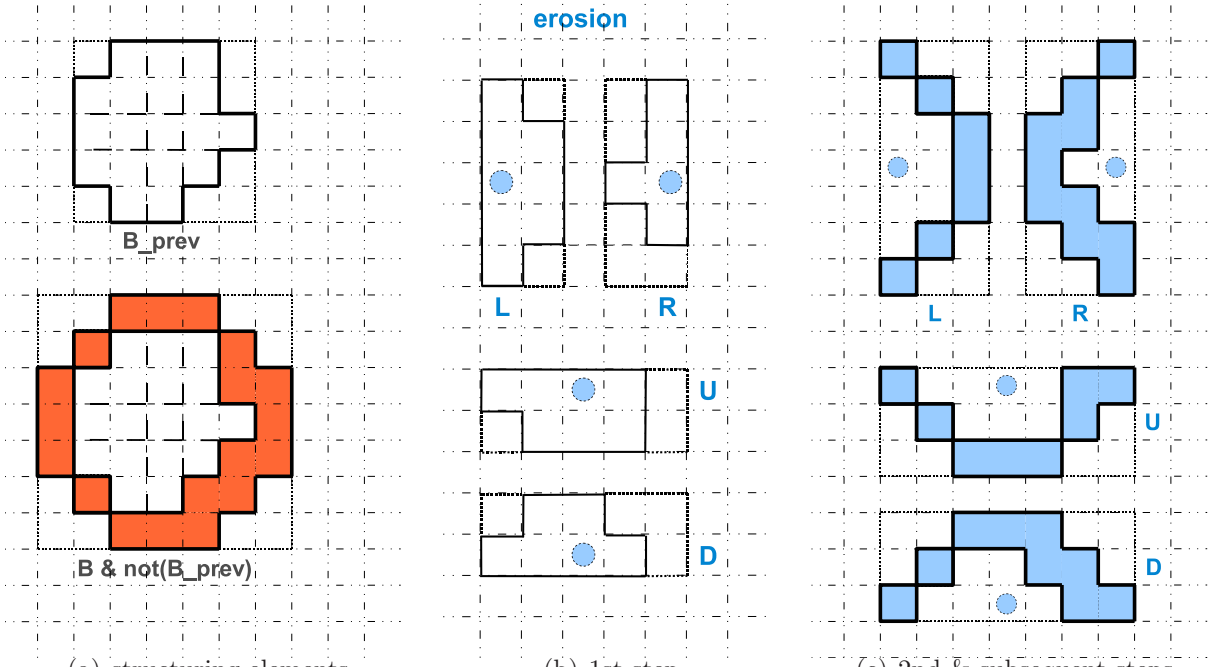
In the same manner this will further dilate `IMG_dilated` with only the pixels present in `B` that weren't already present in `B_prev`, always re-using the `Contours`.

Note that the *edge-effect* is now an option when calling the function in 'erode' mode (that is the objects can be assumed to continue or not beyond the image limits and the erosion to be carried accordingly).

Example of iterative dilation. First generate B as a cell array of boolean masks of increasing size.

```
% first call
[IMG_OUT, Contours]= bmorph(IMG, B{1});
% subsequent calls
for i = 2:size(B,2)
  IMG_OUT = bmorph(IMG_OUT, {B{i}, B{i-1}});
  % insert here code to process or store IMG_OUT
end
```

2.2 Algorithm description and graphical view



(a) structuring elements (b) 1st step (c) 2nd & subsequent steps

Figure 1: iterative algorithm, erosion

A sketch of the erosion algorithm is presented in figure 1. First, the four 1-pixel width contours are detected (left, right, up, down). Then the structuring element is partitioned vertically and horizontally (as per figure 1, b) in order to obtain the L,R and the U,D *'halves'*, respectively. Next, each *'half'* is swept along the corresponding contour, aligning its *'center'* with each contour pixel, and at each alignment the covered pixels are reset.

The iterative extension will take advantage of the already calculated points in repeated calls. At each subsequent erosion step the L/R/U/D *halves* will contain only with the pixels present in `B` that weren't already present in `B_prev` (in other words a *'delta'* structuring element is partitioned as described at 1st step; this delta element can be written as `B & not(B_prev)`). Then the sweep and pixel reset are carried as at first call.

The dilation is done similarly, taking care in *halves* assignment and using pixel set (as opposed to reset, when eroding).

3 Performance

For test a 2304x3456 image is used. In all the figures the performance of `imerode.m`/`imdilate.m` is shown in red and of `bmorph.m` in blue.

The performance improvement when using perfectly circular structuring elements(see Fig. 3) is more than an order of magnitude only when calculating one transform. Of course, this will compound when sweeping.

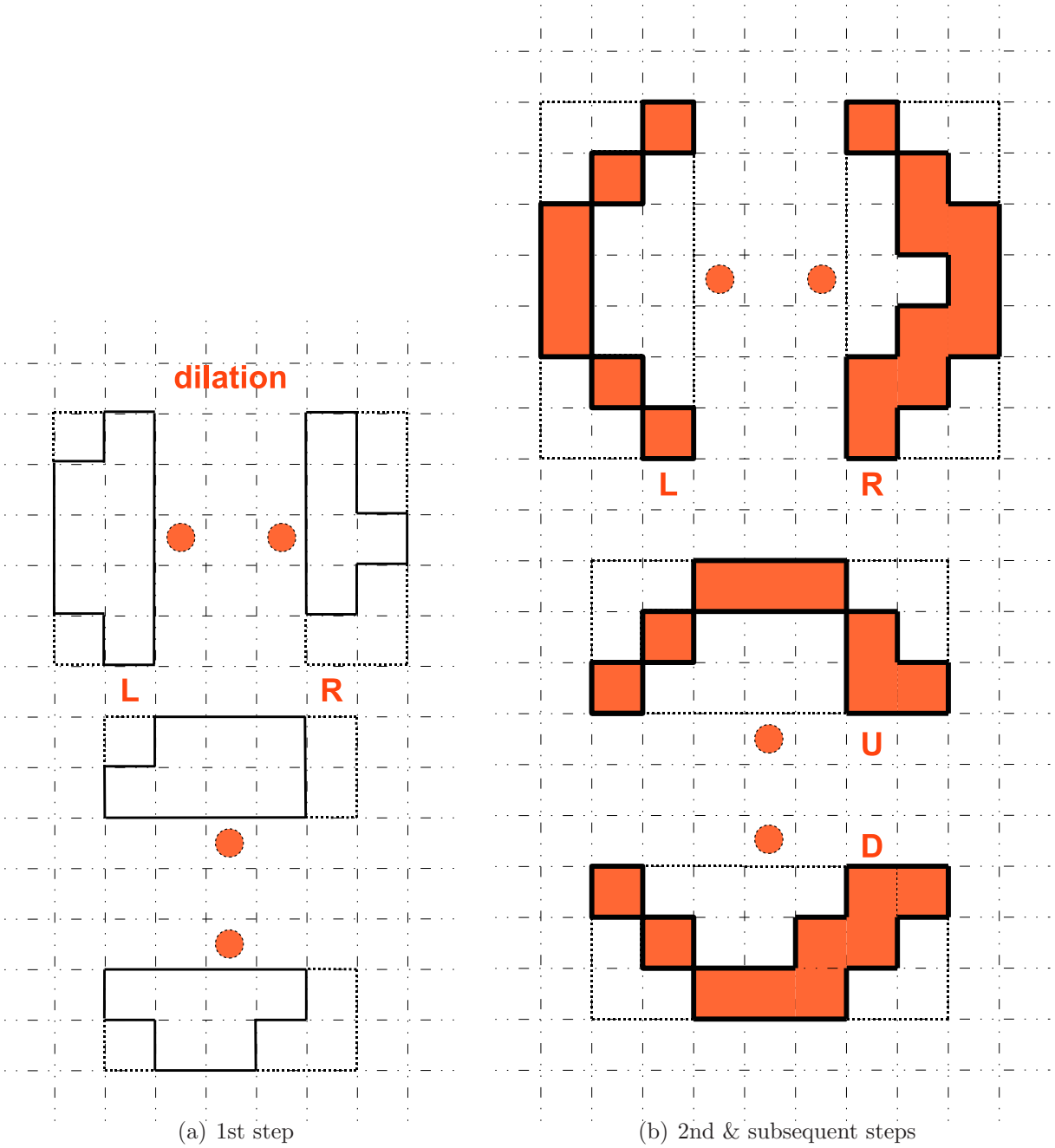
For higher dimensions of the structuring element (75 pixels in this example) this difference increases.

When using 'square' and 'disk' (*i.e.* octagonal) structuring elements `bmorph.m` is slower than `imerode.m` and `imdilate.m` (this might be due to some straight-line optimization inside those functions or simply to data contiguity in the cache).

The performance ratio is about 3 for strels of size 5. This ratio increases to 5, 13 and 24 for strel sizes of 25, 50 and 75 respectively (see Fig. 4 and 5). This means that `bmorph.m` will yield faster results for any sweep longer than 3, 5, 13 and 24 when around the respective strel sizes.

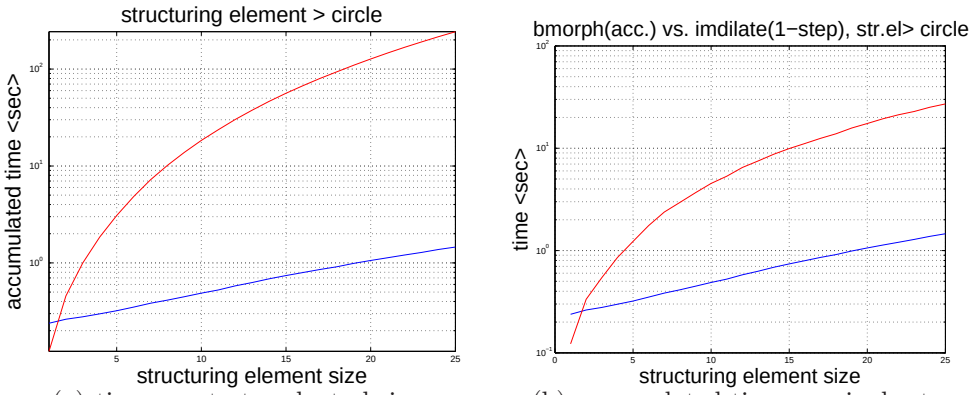
For 'disk' (*i.e.* octagonal) shape the numbers are 3, 10, 29 and 54 for strel sweeps starting at values of 5, 25, 50 and 75 pixels, respectively (see Fig. 6 and 7).

Further optimization could be done by calculating only the 'output contour', then filling in the area between it and the 'input contour'.



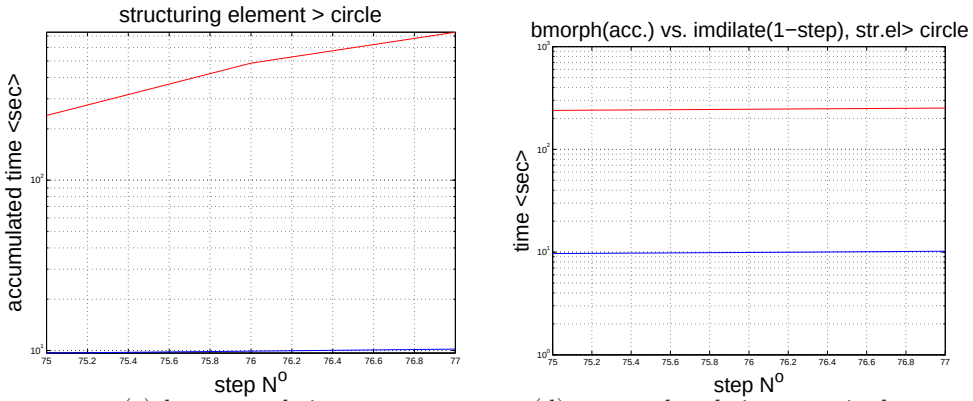
(a) 1st step (b) 2nd & subsequent steps

Figure 2: iterative dilation



(a) time spent at each strel size

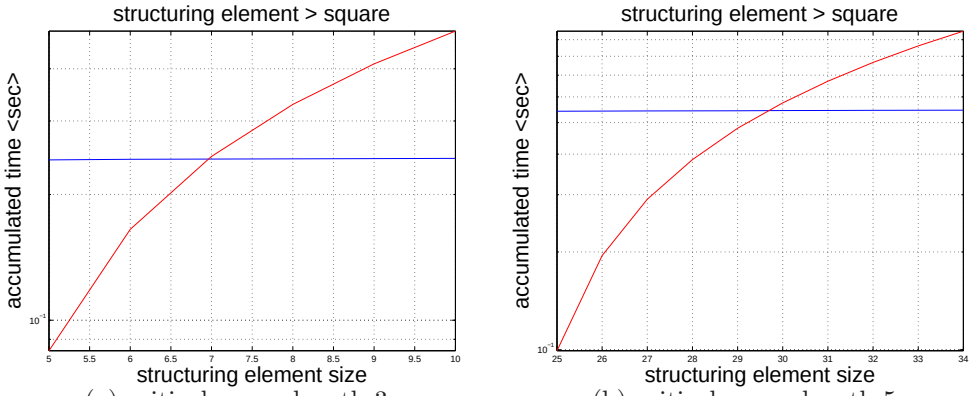
(b) accumulated time vs. single step



(c) larger strel sizes

(d) accumulated time vs. single step

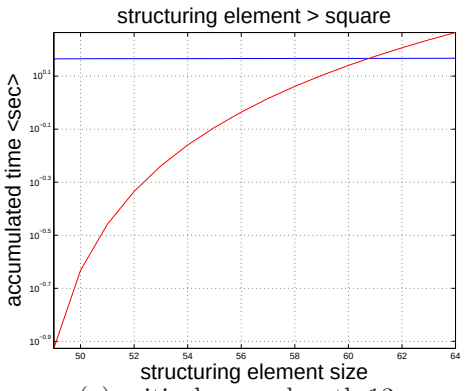
Figure 3: comparison for circular element



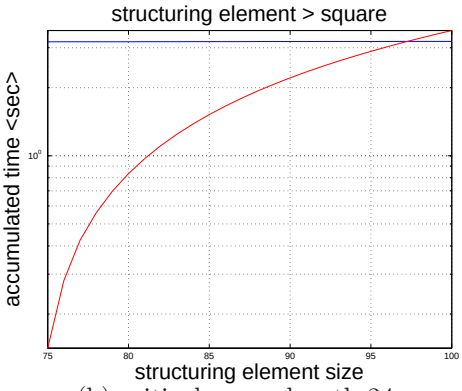
(a) critical sweep length 3

(b) critical sweep length 5

Figure 4: comparison for 'square' element (small strel sizes)

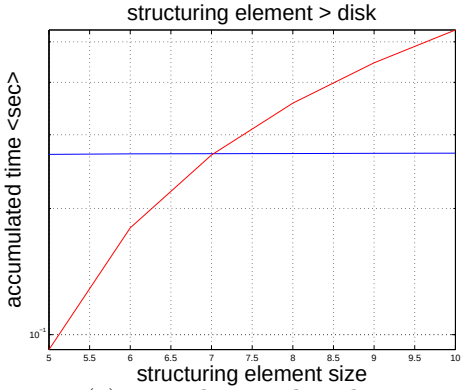


(a) critical sweep length 13

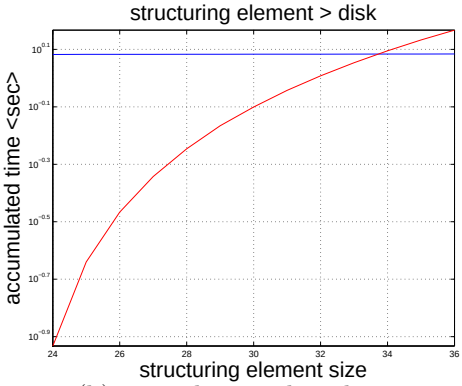


(b) critical sweep length 24

Figure 5: comparison for 'square' element (large strel sizes)

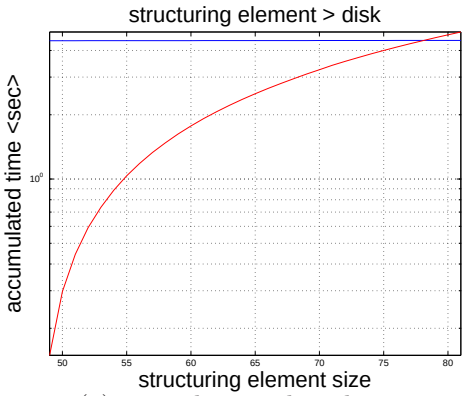


(a) critical sweep length 3

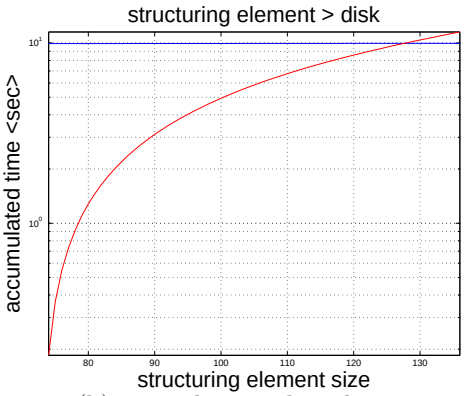


(b) critical sweep length 10

Figure 6: comparison for 'disk' element (small strel sizes)



(a) critical sweep length 29



(b) critical sweep length 54

Figure 7: comparison for 'disk' element (large strel sizes)